Révision automatique des connaissances guidant l'exploration informée d'arbres d'états : application au contexte de la généralisation de données géographiques

Patrick Taillandier

UNIVERSITÉ PARIS-EST

Université Paris-Est
Ecole doctorale ICMS

Thèse

pour obtenir le grade de Docteur de l'Université Paris-Est

Spécialité : Informatique

Patrick TAILLANDIER

Révision automatique des connaissances guidant l'exploration informée d'arbres d'états

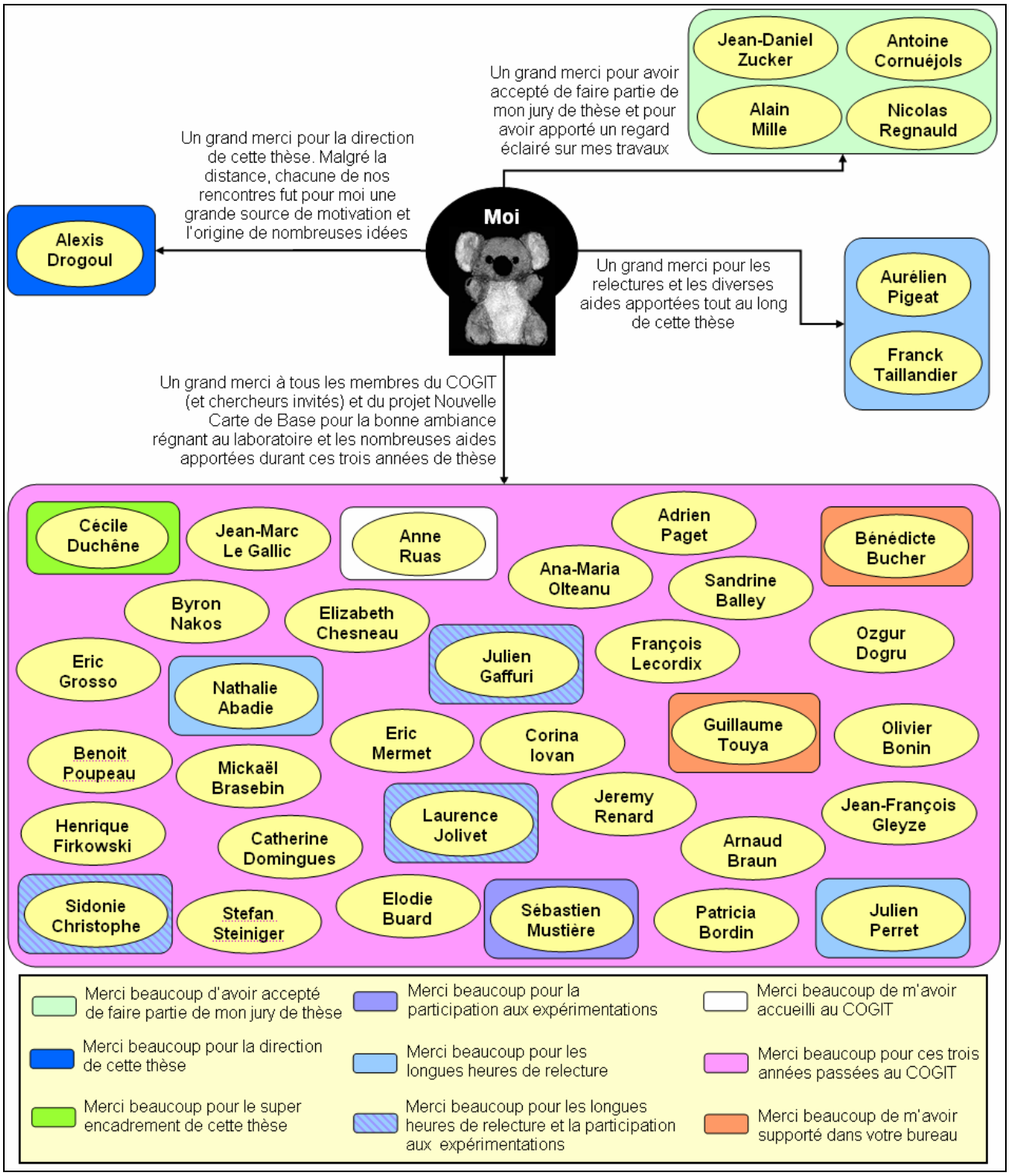
Application au contexte de la généralisation de données géographiques

Soutenue le 2 décembre 2008 devant le jury composé de :

Pr. Antoine Cornuéjols	Rapporteur
Pr. Alexis Drogoul	Directeur
Dr. Cécile Duchêne	Encadrante
Pr. Alain Mille	Examinateur
Dr. Nicolas Regnauld	Examinateur
Pr. Jean-Daniel Zucker	Rapporteur

Remerciements

Parce que parfois un beau dessin vaut mieux qu'un long discours…



Résumé

Cette thèse traite de la révision automatique des connaissances contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états. Ces systèmes, de par leur performance, sont employés dans de nombreux domaines applicatifs. En particulier, des travaux ont proposés d'utiliser cette approche dans le cadre de l'automatisation de la généralisation de données géographiques. La généralisation de données géographique s'intéresse à la dérivation, à partir de données géographiques détaillées, de données moins détaillées adaptées à un besoin particulier (e.g. changement d'échelle). Son automatisation, enjeu majeur pour les agences cartographiques telles que l'Institut Géographique National (IGN), est particulièrement complexe.

Les performances des systèmes basés sur l'exploration informée d'arbres d'états sont directement dépendantes de la qualité de leurs connaissances (heuristiques). Or, la définition et la mise à jour de ces dernières s'avèrent généralement fastidieuses.

Dans le cadre de cette thèse, nous proposons une approche de révision hors ligne des connaissances basée sur le traçage du système et sur l'analyse de ces traces. Ces traces sont ainsi utilisées par un module de révision qui est chargé d'explorer l'espace des connaissances possibles et d'en modifier en conséquence les connaissances du système. Des outils de diagnostic en ligne de la qualité des connaissances permettent de déterminer quand déclencher le processus de révision hors ligne des connaissances. Pour chaque méthode et approche que nous présentons, une mise en œuvre est détaillée et expérimentée dans le cadre de l'automatisation de la généralisation de données géographiques.

Mots clefs : révision des connaissances, exploration informée d'arbres d'états, diagnostic de la qualité des connaissances, généralisation de données géographiques.

Abstract

This work deals with automatic knowledge revision in systems based on an informed tree search strategy. Because of their efficiency, these systems are used in numerous fields. In particular, some literature work uses this approach for the automation of geographic data generalisation. Geographic data generalisation is the process that derives data adapted to specific needs (e.g. map scale) from too detailed geographic data. Its automation, which is a major issue for national mapping agencies like Institut Géographique National (IGN), is particularly complex.

The performances of systems based on informed tree search are directly dependant on their knowledge (heuristics) quality. Unfortunately, most of the time, knowledge definition and update are fastidious.

In this work, we propose an offline knowledge revision approach based on the system logging and on the analysis of these logs. Thus, the logs are used by a revision module which is in charge of the system knowledge revision by knowledge space exploration. Tools for online knowledge quality diagnosis allow to determine when the offline knowledge process should be activated. For each method and each approach presented, an implementation is proposed in the context of geographic data generalisation.

Keywords: knowledge revision, informed tree search strategy, quality knowledge diagnosis, geographic data generalisation.

Table des matières

Liste des figures

Introduction

Contexte général

Exploration informée d'arbres d'états

L'Homme est tous les jours confrontés à des difficultés qui sollicitent son intelligence. Ces difficultés, souvent regroupées sous le terme de problèmes, ont intéressé les chercheurs en intelligence artificielle dès la naissance de la discipline en 1956. Le premier programme d'intelligence artificielle, le Logic Theorist de Newell, Simon et Shaw (Newell & Simon, 1956) était ainsi destiné à la résolution d'un problème particulier : la démonstration de théorèmes. Ce programme proposait de formaliser ce problème sous la forme d'une recherche, à partir de l'hypothèse initiale, de la série d'opérations élémentaires à appliquer pour arriver à la conclusion souhaitée. On utilise souvent une structure arborescente (un arbre d'états) pour représenter cette recherche : l'hypothèse initiale représente la racine de l'arbre et chaque opération élémentaire pouvant être appliquée sur cette hypothèse et amenant à une nouvelle version de cette hypothèse représente une branche de l'arbre. Les nouvelles opérations élémentaires pouvant être appliquées à partir de chacune des nouvelles versions de l'hypothèse initiale forment les nouvelles générations de branches, et ainsi de suite.
La figure 1 donne un exemple d'arbre d'états.

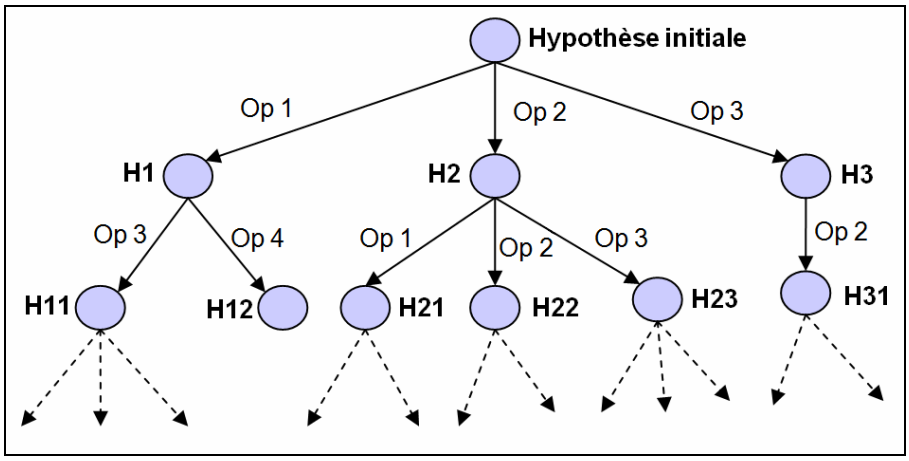


Figure 1 Exemple d'arbre d'états

La conclusion que l'on cherche à démontrer se trouve dans l'arbre. Le problème de la démonstration d'un théorème consiste alors à trouver la série d'opérations permettant, à partir de la racine de l'arbre (l'hypothèse initiale), d'atteindre le nœud souhaité (celui correspondant à la conclusion que l'on cherche à démontrer).
Logic Theorist proposait pour résoudre ce problème d'utiliser une recherche sélective : le choix de l'opération à appliquer à chaque étape (chaque nœud de l'arbre) était guidé par un ensemble de règles empiriques : les *heuristiques*. Ce type de stratégie de résolution de problèmes est communément appelé stratégie d'exploration informée (utilisation *d'heuristiques*) d'arbres d'états. Dans cette thèse, nous regrouperons sous le terme de « *connaissances procédurales* » toutes les connaissances (*heuristiques*) ayant trait à la stratégie d'exploration des arbres d'états.

Aujourd'hui, l'exploration informée d'arbres d'états est couramment employée pour la résolution de nombreux problèmes réels. Un domaine où ce type d'approche a été employé avec succès est celui de l'automatisation de la généralisation de données géographiques. Ce

domaine est particulièrement important pour les agences cartographiques nationales telles que l'Institut Géographique National (IGN) français où s'est déroulé cette thèse.

Généralisation de données géographiques

La généralisation de données géographiques est un processus de simplification de données qui vise à diminuer le niveau de détails de données géographiques afin de les adapter à un besoin particulier. Un cas particulier est la généralisation à but cartographique dont l'objectif est de fournir des données destinées à être affichées de manière lisible à l'écran ou imprimées sur papier à une échelle donnée. La figure 2 donne un exemple de généralisation cartographique : comme le montre la figure, elle ne se limite pas à une simple réduction de la taille des dessins sur la carte mais nécessite l'application de nombreuses opérations telles que des grossissements, des déplacements, ou des éliminations d'objets afin de garantir une bonne lisibilité de la carte tout en gardant l'information essentielle de la carte d'origine.

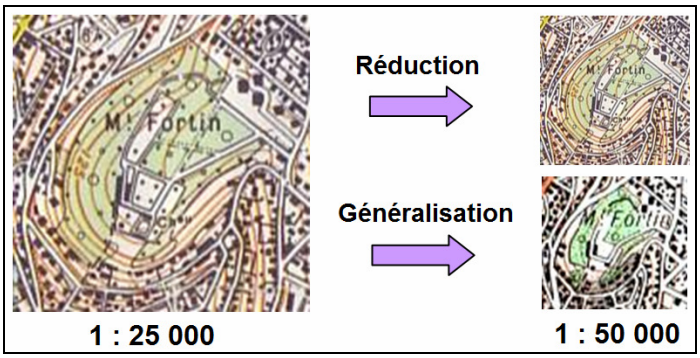


Figure 2 Exemple de généralisation cartographique

L'automatisation de la généralisation, qui est particulièrement complexe, est un enjeu majeur pour les agences cartographiques nationales telles que l'IGN. Elle prend également une place de plus en plus importante grâce à la multiplication des outils de créations de cartes personnalisées notamment sur Internet.

L'une des approches utilisées pour résoudre ce problème d'automatisation consiste à le modéliser sous la forme d'un problème d'exploration d'arbres d'états où le passage d'un état à un autre correspond à l'application d'une opération élémentaire de généralisation sur un ou plusieurs objets géographiques. Cette exploration est guidée par des connaissances permettant au système de généralisation de trouver plus facilement un état satisfaisant au mieux des contraintes cartographiques portant sur les objets géographiques. Généraliser des données géographiques implique généralement d'appliquer des séquences d'opérations élémentaires de généralisation, coûteuses en temps de calcul, sur un grand nombre d'objets géographiques. Il est donc très important que les connaissances du domaine utilisées pour guider le processus de généralisation soient robustes et pertinentes afin de garantir l'obtention d'un résultat de généralisation satisfaisant en un temps acceptable. Malheureusement, définir de telles connaissances se révèle souvent délicat et fastidieux. Il est, de plus, nécessaire de réadapter les connaissances à chaque évolution du système. La problématique de la révision des connaissances procédurales dans le cadre de l'automatisation de la généralisation de données géographiques est donc un enjeu très important.

Objectifs de la thèse

La révision des connaissances est un processus complexe. Elle pose le problème de l'intégration de nouvelles informations dans des bases de connaissances déjà définies. Ce problème se révèle capital pour les systèmes basés sur l'utilisation de bases de connaissances tels que ceux fonctionnant par exploration informée d'arbres d'états. En effet, la définition de bases de connaissances est souvent fastidieuse. Il est souvent complexe de formaliser les connaissances des experts dans un formalisme utilisable par une machine. Eward Feigenbaum a le premier évoqué ce problème et l'a formulé sous le terme de « goulot d'étranglement de l'acquisition des connaissances » (Feigenbaum, 1977). Une approche pour élargir ce goulot est d'utiliser les techniques de l'apprentissage artificiel supervisé. Ces techniques cherchent à construire automatiquement, à partir d'exemples fournis par un expert, un modèle représentant une partie des connaissances de l'expert. Malheureusement, la collecte d'exemples est un processus souvent long. De plus, elle nécessite la participation active d'experts. Une approche pour éviter ce travail de collecte consiste à acquérir directement les exemples par l'expérimentation. Les systèmes basés sur ce type d'approche cherchent à analyser les succès et les échecs connus par le passé pour en déduire des connaissances.

L'objectif de ce travail de thèse est de proposer une approche de révision automatique des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états. Le processus de révision se basera sur l'analyse de traces d'exécution pour réviser automatiquement les connaissances sans avoir à recourir à l'intervention d'experts.

Nous posons également un objectif applicatif : notre approche de révision automatique des connaissances doit permettre la révision automatique des connaissances procédurales contenues dans les systèmes de généralisation basés sur l'exploration informée d'arbres d'états. Nous nous intéresserons particulièrement à la révision des connaissances procédurales du modèle de généralisation AGENT (Ruas, 1999 ; Lamy et al., 1999 ; Barrault et al., 2001).

Plan

Le chapitre A introduit le contexte applicatif dans lequel se place ce travail de thèse, celui de l'automatisation de la généralisation de données géographiques. Nous présentons d'abord la généralisation de données géographiques ainsi que quelques notions essentielles liées l'information géographique. Nous proposons ensuite un aperçu des travaux visant à automatiser la généralisation. Nous détaillons plus particulièrement le modèle proposé par (Ruas, 1999) et repris dans le projet européen AGENT. Ce modèle est basé sur l'exploration informée d'arbres d'états. Il servira de cadre applicatif à nos travaux. Nous insistons enfin sur les enjeux de la révision des connaissances pour un tel modèle de généralisation automatique. Nous concluons ce chapitre en exposant l'objectif applicatif de cette thèse et en le généralisant à tout système fonctionnant par exploration informée d'arbres d'états. Cette généralisation nous permet d'introduire l'objectif scientifique de cette thèse.

Le chapitre B présente les problématiques soulevées par l'objectif de cette thèse et introduit l'approche générale retenue. Nous proposons d'abord un état de l'art des travaux s'intéressant à l'acquisition et à la révision des connaissances en intelligence artificielle ainsi qu'à leur application dans le cadre de la généralisation automatique de données géographiques. Nous proposons ensuite une première formalisation du problème que nous cherchons à résoudre dans cette thèse. Cette formalisation nous permet ensuite d'analyser les problématiques soulevées par l'objectif de cette thèse. Nous présentons dans une dernière partie notre

approche générale ainsi que notre modèle général. Notre approche générale est une approche hors ligne. Elle est basée sur le traçage du processus de résolution de problèmes pendant la résolution d'un échantillon représentatif d'instances du problème et sur l'utilisation de ces traces pour réviser les connaissances. Notre modèle est basé sur l'externalisation des connaissances et sur leur agentification. En vue de remplir notre objectif applicatif, nous présentons comment appliquer cette approche et ce modèle généraux au cadre du modèle AGENT.

Le chapitre C présente la phase de constitution de l'expérience. Celle-ci consiste à tracer le processus de résolution de problèmes pendant la résolution d'un échantillon représentatif d'instances du problème. Nous proposons d'abord une approche visant à sélectionner un échantillon représentatif parmi un ensemble d'instances du problème. Nous nous intéressons ensuite au problème du choix du jeu de connaissances à utiliser lors du traçage du processus. Nous proposons enfin une approche menée au niveau de chaque connaissance et visant à construire des informations supplémentaires à partir d'une analyse locale des instances résolues.

Le chapitre D présente la phase de révision des connaissances. Nous proposons, dans ce chapitre, des approches visant à utiliser l'expérience acquise (les traces) en vue de réviser les connaissances du système de résolution de problèmes. Nous présentons d'abord les problématiques soulevées par cette phase de révision. Nous présentons ensuite une approche générale de révision basée sur la décomposition de l'ensemble des connaissances en groupes de connaissances et sur la révision séquentielle de chaque groupe de connaissances. Nous nous intéressons enfin à la révision de chacun de ces groupes de connaissances. Nous proposons ainsi une approche générique de révision des connaissances basée sur l'exploration du domaine de définition des connaissances. Nous présentons, dans ce cadre, un court état de l'art des méthodes d'exploration en intelligence artificielle. Cette approche générique peut se révéler inefficace dans le cadre de connaissances complexes. Des approches plus spécialisées sont alors nécessaires : nous proposons ainsi une approche destinée à la révision des connaissances représentées sous forme de bases de règles de production.

Le chapitre E s'intéresse au diagnostic en ligne de la qualité des connaissances. Nous présentons d'abord les principales difficultés liées à ce diagnostic. Nous proposons ensuite une approche de diagnostic visant à répondre à ces difficultés. Cette approche est basée sur une analyse, pour chaque instance résolue du problème, des succès et des échecs connus par chaque connaissance ainsi que sur une analyse globale de la qualité des résultats obtenus. Elle utilise enfin une méthode de décision multicritère pour déterminer s'il est nécessaire ou non de déclencher le processus de révision des connaissances.

Le chapitre F est consacré à l'évaluation de l'ensemble des approches et des méthodes que nous avons proposées dans les chapitres B, C, D et E. Nous présentons ainsi un ensemble d'expérimentations menées dans le cadre de la révision des connaissances du modèle de généralisation AGENT. Nous proposons une analyse détaillées des résultats obtenus. Nous exposons enfin les difficultés liées à la mise en œuvre de nos approches dans le cadre d'un système de résolution de problèmes donné et nous exposons quelques pistes pour répondre à celles-ci.

Remarques sur le vocabulaire et les notations

Certains termes, tels que celui de *généralisation*, ont un sens différent en géomatique et en intelligence artificielle. Nous avons essayé d'éviter toute confusion dans le texte, tout en conservant les termes consacrés. Nous proposons, en fin de mémoire, un glossaire précisant le sens donné à chacun de ces termes.

Chapitre A
Contexte applicatif : la généralisation de données géographiques

A.I Introduction

La résolution de problèmes complexes a toujours fait partie des thèmes centraux de l'intelligence artificielle. Parmi les méthodes de résolution figurent celles basées sur des stratégies d'exploration informée (heuristiques). Ces méthodes, qui tirent parti de connaissances spécifiques au problème, se révèlent dans la plupart des cas bien plus efficaces que les méthodes d'exploration non informée. Cette efficacité explique leur succès dans de nombreux domaines. L'un de ces domaines est celui de la généralisation automatique de données géographiques.

La généralisation de données géographiques est le processus qui consiste à dériver à partir de données géographiques détaillées, des données moins détaillées adaptées à un besoin particulier. Son automatisation, qui est un enjeu majeur pour les agences cartographiques nationales telles que l'Institut Géographique National (IGN) français, est particulièrement complexe. De nombreux travaux de recherche ont porté sur ce problème. Parmi eux, certains se basent sur une exploration informée d'arbres d'états où le passage d'un état à un autre correspond à l'application d'une opération élémentaire de généralisation sur un ou plusieurs objets géographiques. Le recours à la généralisation automatique dans le cadre d'applications réelles implique souvent de traiter un grand nombre d'objets géographiques et d'utiliser des opérations élémentaires de généralisation qui peuvent s'avérer complexes en termes de temps de calcul. Il est donc très important que les connaissances du domaine utilisées pour guider le processus de généralisation soient robustes et pertinentes afin de garantir l'obtention d'un résultat de généralisation satisfaisant en un temps acceptable. Malheureusement, la définition des connaissances se révèle généralement délicate et fastidieuse. Il est, de plus, nécessaire de réadapter les connaissances à chaque évolution du système telle que l'ajout d'un nouvel élément (ex : une nouvelle opération de généralisation). Il est donc particulièrement intéressant de disposer de méthodes permettant de réviser automatiquement ces connaissances.

La problématique de la révision des connaissances dans le cadre de l'automatisation de la généralisation de données géographiques est donc un vrai enjeu et une application particulièrement intéressante pour les approches de révision des connaissances. Ce travail de thèse s'inscrit dans cette problématique. Notre objectif applicatif est ainsi de proposer un modèle de révision automatique des connaissances destiné aux systèmes de généralisation basés sur l'exploration informée d'arbres d'états.

Dans ce premier chapitre, nous reviendrons plus en détail sur la notion de généralisation de données géographiques (A.II). Nous présenterons brièvement les travaux visant à son automatisation (A.III) et nous détaillerons le modèle de généralisation AGENT (A.IV). Ce dernier, qui est utilisé aujourd'hui en production à l'IGN, et qui est basé sur l'exploration informée d'arbres d'états, constitue le cadre applicatif de nos travaux.

Nous reviendrons également, dans ce premier chapitre, sur les difficultés posées par les systèmes de généralisation basés sur l'exploration informée d'arbres d'états, et en particulier sur celles provenant de la définition des connaissances du domaine et de leur évolutivité (A.V).

L'introduction de ce contexte applicatif nous permettra, dans une dernière partie (A.VI), de généraliser ce problème de révision de connaissances à tous les systèmes basés sur l'exploration informée d'arbres, et plus seulement à ceux dédiés à la généralisation de données géographiques. Nous présenterons enfin, dans cette dernière partie, les objectifs applicatifs et scientifiques de cette thèse.

A.II La généralisation de données géographiques

A.II.1 Représentation numérique de l'information géographique

Nous allons introduire, dans cette première partie, quelques notions relatives à la modélisation des données géographiques et à l'information géographique.

A.II.1.1 Modes de représentation des données géographiques

Il existe deux grands modes de représentation des données géographiques sous forme numérique : les représentations *vecteur* et *raster* (figure A.1).

Le *mode raster* (ou maillé) se base sur le partitionnement de l'espace en un maillage constitué de cellules rectangulaires (et généralement carrées) appelées pixels. Chaque pixel porte une ou plusieurs informations descriptives caractérisant son contenu. La géométrie d'un objet géographique correspond à un ensemble de pixels. Ce mode de représentation est utilisé pour les photographies numériques.

Le *mode vecteur* consiste à représenter la géométrie des objets géographiques sous la forme d'une primitive géométrique, qui peut être, en deux dimensions, un point (décrit par ses coordonnées dans un repère), une ligne (décrite par une suite de points), ou une surface (décrite par son contour qui est une ligne fermée). Les points constitutifs des lignes peuvent être reliés entre eux par des segments ou par des courbes plus complexes telles que des splines ou des courbes de Béziers. Ce mode de représentation est utilisé par la plupart des logiciels de CAO/DAO.

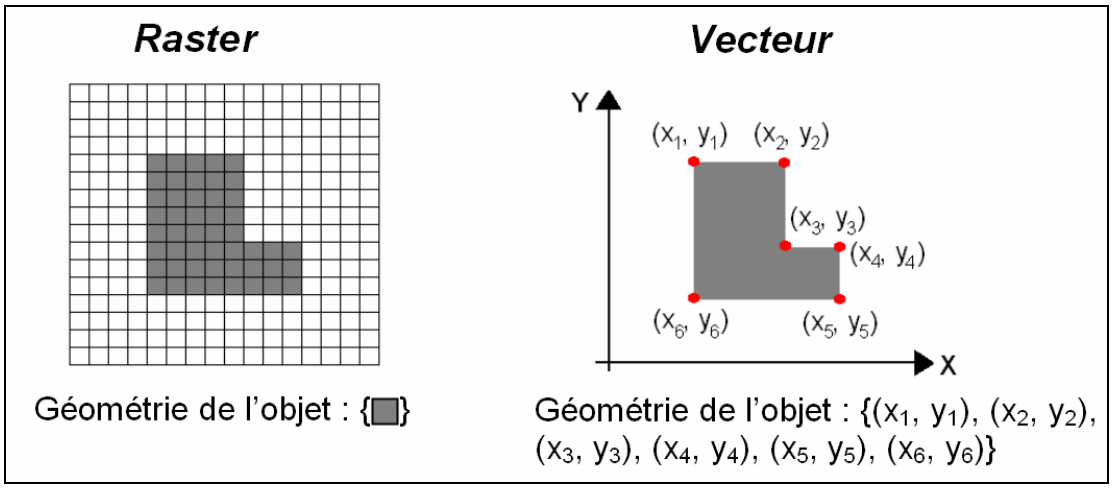


Figure A.1 Exemple de représentation raster et vecteur de la géométrie d'un bâtiment

Dans le cadre de cette thèse, nous nous intéresserons uniquement aux données géographiques représentées sous forme vectorielle. Nous désignons donc par « base de données géographique », une base de données géographique vectorielle.

A.II.1.2 Base de données géographique

Une base de données géographique est constituée d'un ensemble d'objets organisés en classes d'objets de même nature. La figure A.2 donne un exemple de données géographiques composées de trois classes d'objets différentes : une classe végétation, une classe bâtiment et une classe route.
Chaque objet de la base est géoréférencé, c'est-à-dire qu'il porte un attribut représentant son emprise spatiale (sa géométrie). Dans le cadre d'une représentation 2D vecteur des géométries, ces dernières sont exprimées sous la forme de primitives points, lignes et surfaces. Un objet peut également disposer, en plus de sa géométrie, d'un ensemble d'attributs similaires à ceux que l'on trouve classiquement dans toute base de données. Ces derniers permettent de compléter la description de l'objet. Par exemple, un tronçon routier peut porter un attribut donnant des informations sur le type de route qu'il représente (départementale, nationale, autoroute, etc.).
Dans la très grande majorité des cas, le type de géométrie (point, ligne ou surface) est défini au niveau de la classe. Tous les objets d'une même classe partagent donc le même type de géométrie.

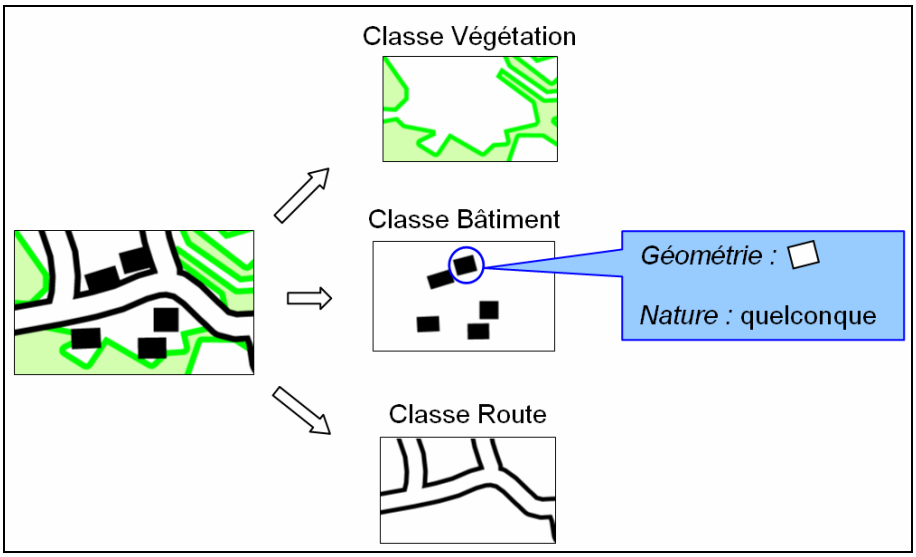


Figure A.2 Organisation en classe de données géographiques

A.II.1.3 Système d'information géographique

Un système d'information géographique (SIG) est un outil informatique permettant de manipuler des données géographiques. Il permet en particulier de gérer des bases de données géographiques, mais aussi de visualiser leur contenu en attribuant des symboles cartographiques aux différents objets de la base. La figure A.3 donne un exemple de visualisation obtenue avec le SIG Clarity™.

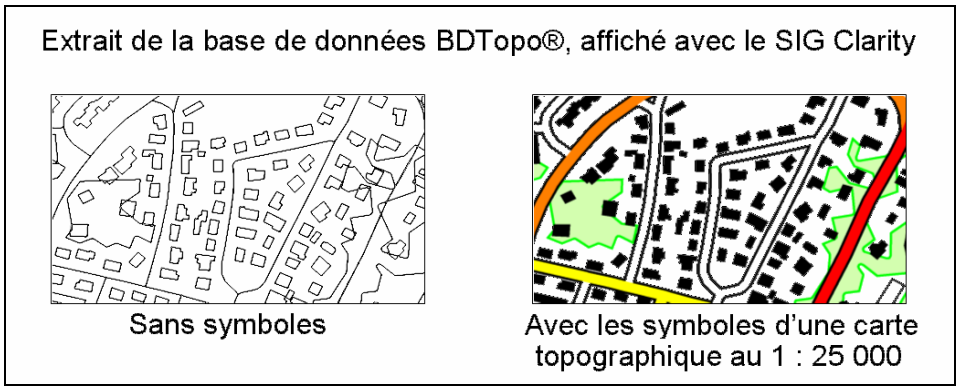


Figure A.3 Extrait de données géographiques affiché par le SIG Clarity

Un SIG dispose d'outils d'analyse et de gestion des données. Il permet ainsi d'effectuer des calculs et des requêtes complexes s'appuyant sur les géométries des objets mais aussi d'appliquer des transformations sur ces dernières. La plupart des SIG proposent un environnement de programmation permettant de directement coder et intégrer de nouveaux éléments.

A.II.2 La généralisation de données géographiques

A.II.2.1 Echelle et niveau de détail

Dans le cadre d'une carte papier, *l'échelle* représente le facteur d'homothétie existant entre la taille des entités sur le terrain et sur la carte. Plus l'échelle d'une carte est grande, plus la taille des objets est importante sur la carte. La figure A.4 présente des exemples de cartes de la même zone à différentes échelles.

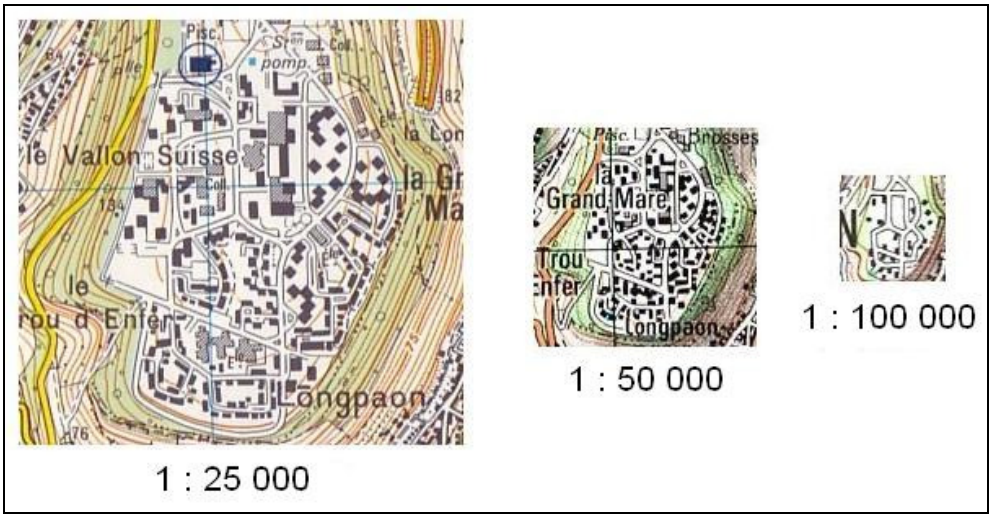


Figure A.4 Extraits de cartes de Rouen à différentes échelles

On ne peut pas directement parler d'échelle pour les bases de données géographiques. En effet, les SIG permettent d'afficher des données géographiques à différentes échelles grâce à leurs outils de visualisation (zoom). Une base de données géographique a, par contre, une *échelle de référence* et un *niveau de détail*.

L'échelle de référence désigne l'échelle la plus adaptée pour la visualisation des données. Le niveau de détail traduit la finesse des données contenues dans la base. Deux types de niveaux de détail ont été distingués par (Ruas & Bianchin, 2002) :

- Le *niveau de détail géométrique* qui correspond à la finesse de description de la géométrie des objets.
- Le *niveau de détail sémantique* qui correspond à la finesse de choix des objets représentés dans la base de données.

A.II.2.2 Généralisation de données géographiques

La généralisation de données géographiques est le processus qui consiste à dériver, à partir de données détaillées, des données moins détaillées et adaptées à un besoin. (Ruas, 2002, p.75) propose une comparaison de ce processus avec celui du résumé de texte :

> « La généralisation est un processus de synthèse d'information. On peut comparer ce processus avec celui d'un résumé de texte dont les objectifs sont de réduire le nombre de mots, de retenir les idées principales, de ne pas faire de faux sens et si possible de conserver le style de l'auteur. Evidemment, le texte doit respecter les règles du langage tant au niveau de l'orthographe que de la grammaire. Pour bien généraliser l'information initiale, il faut également réduire la quantité d'information, mettre en valeur l'information la plus importante, rester fidèle à l'information initiale et, dans le cas d'une carte, respecter les règles de sémiologie qui permettent une bonne lecture de l'information ».

Deux types d'opérations sont utilisés pour la généralisation :

- Les opérations qui agissent sur le schéma de données : suppression d'une classe, fusion de classes, fusion de valeurs d'attributs, etc.
- Les opérations qui agissent sur les objets : simplification de la forme d'un objet, suppression d'objets au sein d'une classe, etc.

A.II.2.3 Généralisation cartographique

La *généralisation à but cartographique*, aussi appelée *généralisation cartographique*, est un type particulier de généralisation de données géographiques qui vise à produire une carte. L'ACI (Association Cartographique Internationale) l'a définie, en 1973, comme « la sélection et la représentation simplifiée de détails en fonction de l'échelle et des objectifs de la carte ».

Avant l'apparition des données numériques, la généralisation cartographique était utilisée pour créer une carte à une échelle donnée à partir d'une autre carte à une échelle plus grande. De nos jours, la généralisation cartographique peut aussi être utilisée pour la création d'une carte à partir d'une base de données géographique trop détaillée par rapport à l'échelle souhaitée. On désigne par *base de données cartographique* la base de données alors dérivée (figure A.5).

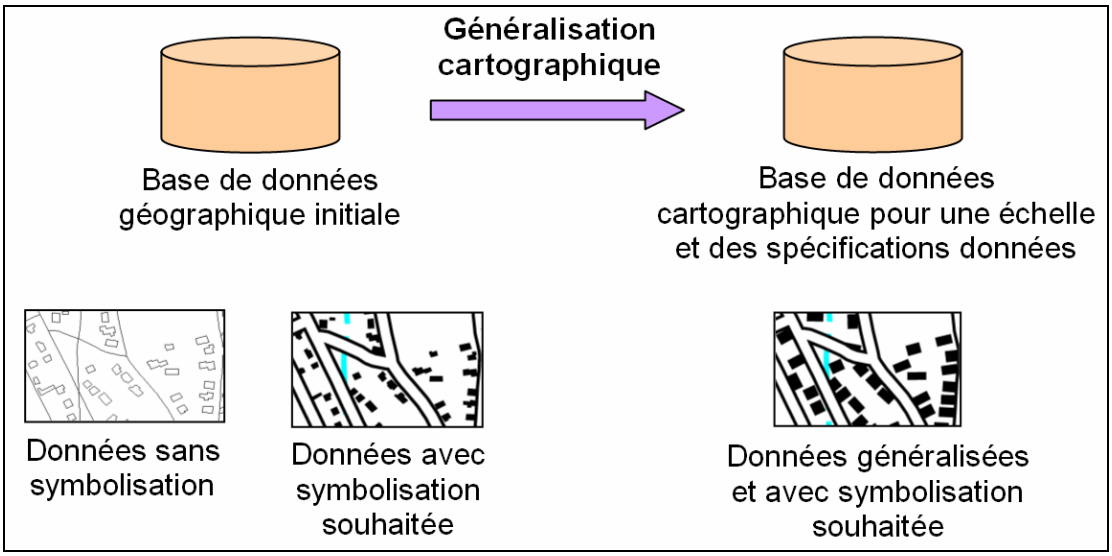


Figure A.5 Généralisation cartographique

Le recours à la généralisation cartographique est rendu nécessaire par les limites de la perception humaine.

Un premier aspect de ces limites concerne les limitations de l'œil humain. En effet, même sans tenir compte des limites de résolution des écrans et des imprimantes, l'œil humain ne peut percevoir des détails trop petits. Ainsi, certains objets trop petits peuvent ne plus être perçus une fois le passage à l'échelle souhaitée accompli. De même, des problèmes de séparation et de différenciation entre objets peuvent apparaître.

Un second aspect des limites de la perception humaine concerne notre difficulté à analyser simultanément de grandes quantités d'informations. Les capacités d'analyse de notre cerveau sont très inférieures à nos capacités visuelles : une carte efficace doit faire ressortir uniquement l'information importante. Une carte est considérée comme difficilement analysable lorsqu'elle contient trop d'éléments, ou lorsqu'elle n'a pas de structure globale claire (Eastman, 1985).

L'objectif de la généralisation cartographique est de rendre lisibles des données géographiques compte tenu des spécifications de la carte définies par l'utilisateur. Une généralisation particulière doit ainsi prendre en compte des éléments tels que l'échelle de la carte et la symbolisation souhaitées, les classes à représenter, les souhaits de l'utilisateur (ex : privilégier la conservation des positions des objets plutôt que leur forme), etc.

A.II.2.4 Contraintes et conflits cartographiques

Comme nous l'avons mentionné dans la partie précédente, la généralisation consiste à appliquer des transformations sur des données géographiques afin que celles-ci soient lisibles tout en conservant les informations importantes des données initiales. Généraliser revient donc à satisfaire un ensemble de *contraintes cartographiques* (Beard, 1991 ; Weibel, 1996 ; Weibel & Dutton, 1998 ; Ruas, 1999) qui peuvent concerner aussi bien des problèmes de lisibilité que des problèmes de conservation de l'information importante. On utilise le terme *conflit cartographique* pour désigner le fait qu'une contrainte cartographique n'est pas respectée.

Il n'est en général pas possible d'éliminer tous les conflits cartographiques. En effet, certaines contraintes cartographiques peuvent s'avérer contradictoires : typiquement, un bâtiment superposé à une route ne respecte pas la contrainte cartographique de non superposition entre les bâtiments et les routes, mais déplacer ce bâtiment peut entraîner la violation d'une autre contrainte cartographique de maintien des positions d'origine.

Généraliser consiste donc à trouver le meilleur compromis possible entre les contraintes, compte tenu des besoins concernant les données dérivées.

A.II.2.5 Opérations de généralisation

Le cartographe dispose, pour généraliser des données géographiques, d'un ensemble d'opérations possibles. De nombreuses classifications de ces opérations de généralisation ont été proposées (McMaster & Shea, 1992 ; Jones, 1997 ; Weibel & Dutton, 1999 ; Ruas, 1999 ; Bader, 2001 ; Li, 2007 ; Foester & Stöter, 2007). Nous présentons ici, celle proposée par (Mustière, 2001, p.29-35) qui divise les opérations de généralisation en trois grandes familles :

- *Les opérations de simplification* (figure A.6) qui visent à réduire la quantité d'informations en éliminant celles peu importantes.
- *Les opérations de caricature* (figure A.7) qui visent à accentuer certains caractères des objets au détriment d'autres.
- *Les opérations d'harmonisation* (figure A.8) qui visent à éliminer les différences entre les objets.

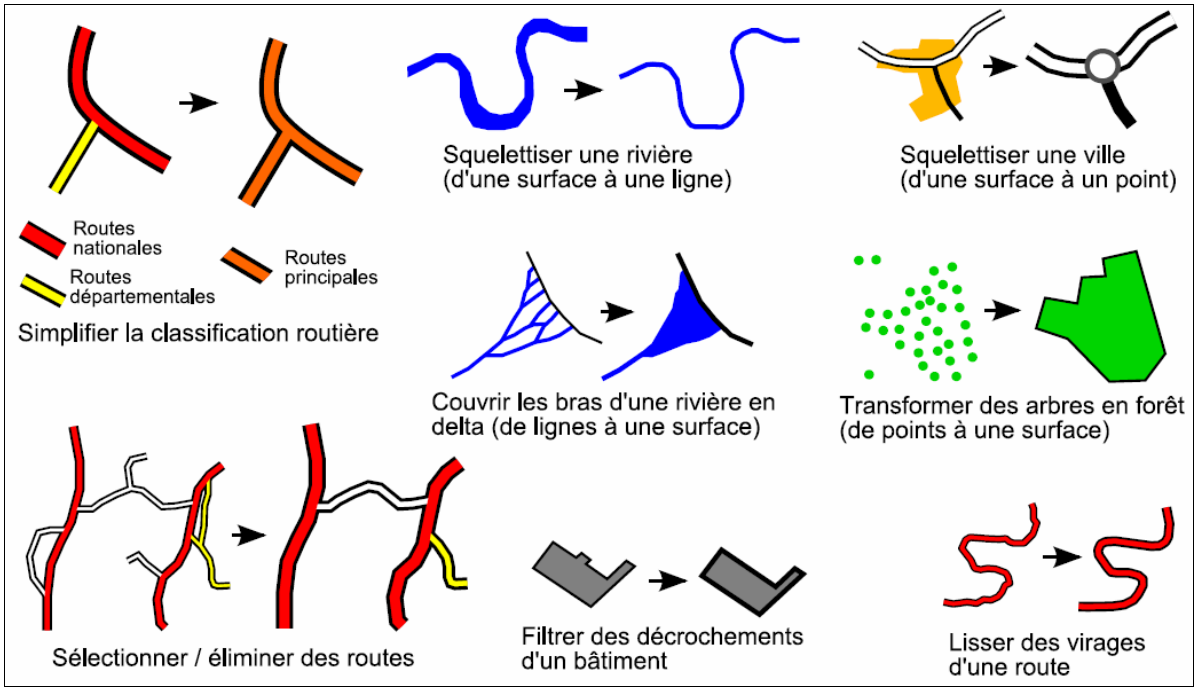


Figure A.6 Exemples d'opérations de simplification (Mustière, 2001, p. 31)

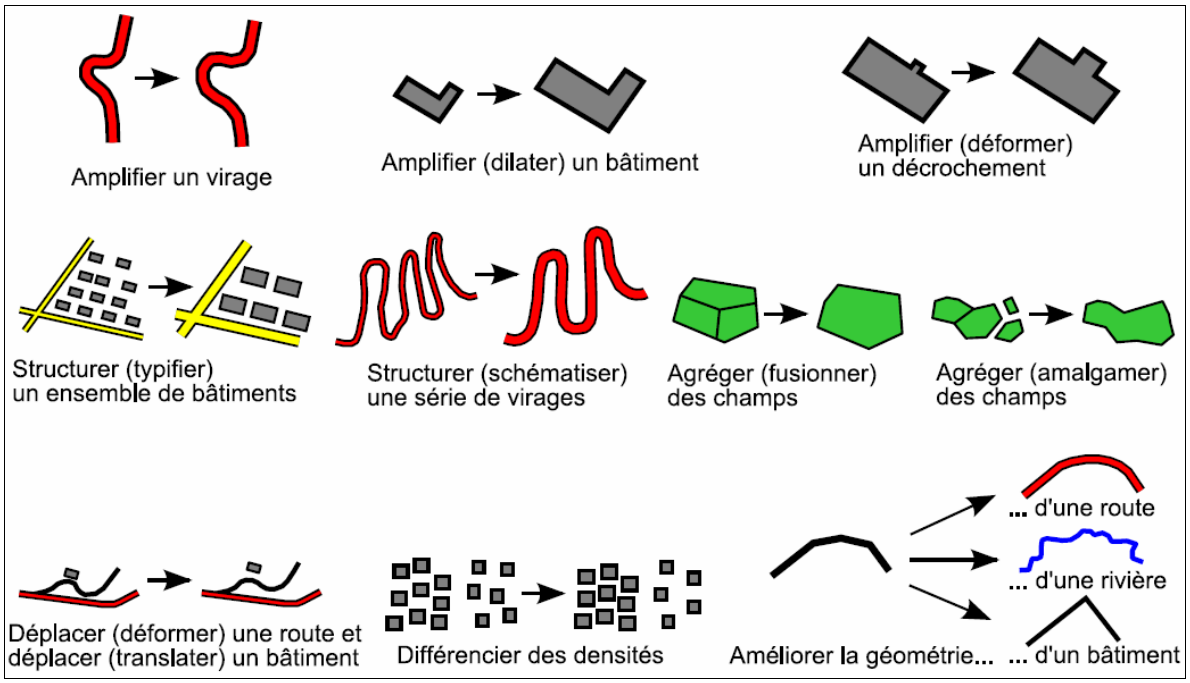


Figure A.7 Exemples d'opérations de caricature (Mustière, 2001, p. 33)

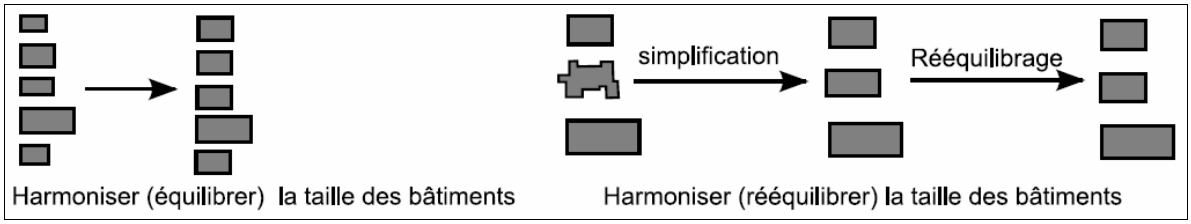


Figure A.8 Exemples d'opérations d'harmonisation (Mustière, 2001, p. 35)

A.III L'automatisation de la généralisation de données géographiques

A.III.1 Enjeu de l'automatisation de la généralisation

La généralisation de données géographiques est un processus complexe. Même assistée par ordinateur, la généralisation interactive reste un processus long et coûteux. Des études menées à l'IGN ont estimé que le temps de généralisation pour réaliser des cartes IGN au 1 : 100 000 et au 1 :1 000 000 à partir de la BDCarto® (base de données de précision décamétrique), était compris entre 6 mois et 2 années-homme de travail par carte (Mustière 2001, p. 19).

En raison de ces temps prohibitifs, les agences de cartographie tendent parfois à entretenir en parallèle plusieurs bases de données adaptées à différents besoins (typiquement à la production de cartes à différentes échelles) ce qui pose le problème de leur entretien. En effet, entretenir plusieurs bases de données en parallèle nécessite de répercuter les mises à jour séparément sur chacune de ces bases, ce qui peut se révéler coûteux à terme. Disposer d'un processus le plus automatique possible de généralisation peut permettre de grandement limiter le temps nécessaire à la généralisation et de pouvoir dériver à partir d'une unique base de données très détaillée (dite de *référence*), toutes les bases de données souhaitées (Ruas, 2002).

A.III.2 Problématique de l'automatisation de la généralisation

A.III.2.1 Algorithmes de généralisation et outils d'analyse spatiale

Les premiers travaux qui ont visé à automatiser la généralisation ont cherché à traduire, sous forme d'algorithmes, les opérations élémentaires de généralisation utilisées par les cartographes (présentées en partie A.II.2.5). De nombreux états de l'art de ces algorithmes ont été proposés, parmi ceux-ci (Mustière & Lecrodix, 2002 ; Regnauld, 2002 ; Galanda & Weibel, 2002 ; Saux & al., 2002 ; Regnauld & McMaster, 2007). La figure A.9 présente des exemples de résultats obtenus avec certains de ces algorithmes.

De nombreux travaux portent encore sur la conception d'algorithmes de généralisation (Yan & Li, 2004 ; Gokgoz, 2005 ; Mackaness & Steven, 2006). D'autres travaux se concentrent sur l'utilisation de ces algorithmes à partir de services web (Burghardt et al., 2005 ; Neun & Burghardt, 2005 ; Foerster & Stöter, 2006).

Disposer d'algorithmes de généralisation n'est en soi pas suffisant pour assurer l'automatisation complète du processus de généralisation. Il faut, en effet, se poser la question de savoir quels algorithmes employer, sur quels objets et avec quels paramétrages (Brassel & Weibel, 1988 ; Shea & McMaster, 1989).

Répondre à ces questions nécessite dans un premier temps, de disposer d'outils d'analyse et plus particulièrement de mesures (Shea & McMaster, 1989) permettant une caractérisation des données à généraliser. Une mesure permet, en effet, d'expliciter les caractéristiques implicites de données géographiques. Par exemple, une mesure peut donner des indications sur la forme d'un bâtiment (Rainsford & Mackaness, 2002), sur la régularité d'un alignement de bâtiments (Ruas & Holzpfel, 2003) ou sur la forme d'un virage (Barillot, 2002a). De

nombreuses mesures ont ainsi été proposées dans la littérature (Uni-Zh, 1999), ainsi que différentes classifications de ces mesures (Barrillot, 2002b ; Mackaness & Ruas, 2007).

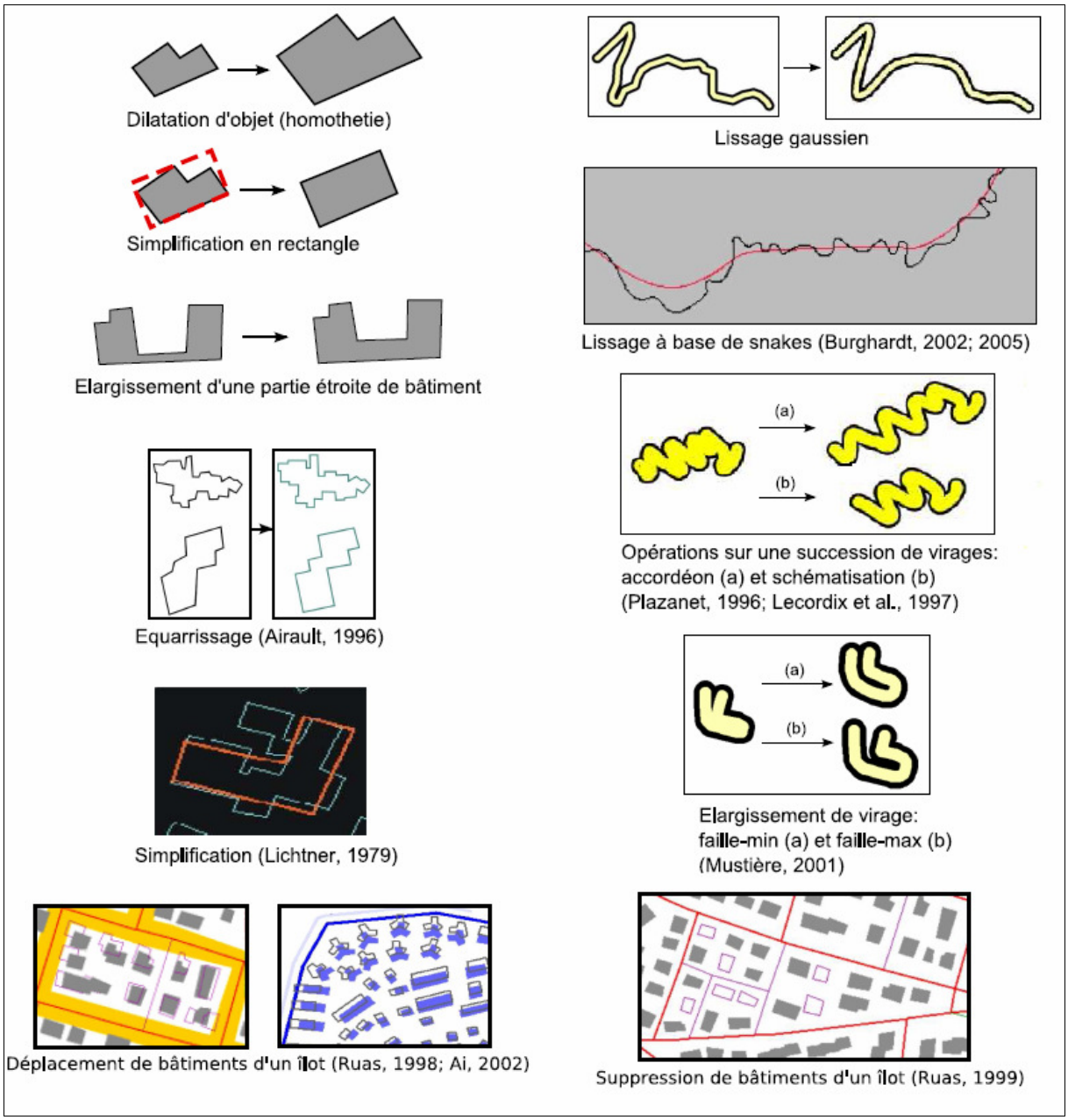


Figure A.9 Exemples de résultats d'algorithmes de généralisation. D'après (Gaffuri, 2008, p. 38)

A.III.2.2 Passage de l'algorithme au processus

Nous avons présenté, dans la partie précédente (A.III.2.1), des algorithmes de généralisation et des outils d'analyse spatiale. Ces éléments servent de briques de base aux processus complets de généralisation. En effet, un processus de généralisation est un algorithme qui a pour charge de généraliser un objet ou un groupe d'objets en appliquant sur ces derniers, en fonction des valeurs prises par les diverses mesures utilisées pour caractériser l'espace géographique, un ensemble d'algorithmes de généralisation.

Les premiers processus de généralisation sont apparus au début des années 1990 (Müller & Wang, 1992 ; Schyllberg, 1992) et concernaient seulement quelques thèmes géographiques. Depuis, de nombreux processus de généralisation, plus génériques et pouvant s'appliquer à différents thèmes, ont été proposés.

Nous présentons, dans les parties suivantes, différentes approches proposées dans la littérature pour l'automatisation de la généralisation. La classification proposée est directement inspirée de (Duchêne, 2004) et de (Gaffuri, 2008).

A.III.3 Séquence prédéfinie

Une première approche d'automatisation de la généralisation consiste à définir statiquement une séquence d'algorithmes pour chaque classe d'objets. Ainsi, chaque objet de la classe concernée se voit appliquer une séquence prédéfinie d'algorithmes. Cette séquence peut tenir compte des valeurs attributaires des objets et ainsi varier d'un objet à un autre au sein d'une même classe.

Plusieurs SIG proposent des outils visant à assister la conception de ces séquences (Lee, 2003 ; Smith, 2003). La figure A.10 donne un exemple de l'un de ces outils, le ModelBuilder proposé par l'éditeur de SIG ESRI. Dans cet exemple, le processus de généralisation de courbes de niveau défini est le suivant : les courbes de niveau équidistantes de 50 m sont sélectionnées. Le processus applique ensuite à ces courbes sélectionnées une opération de filtrage. Enfin, le processus détruit les courbes qui ont été transformées en un point par l'opération de filtrage et effectue un lissage sur les autres.

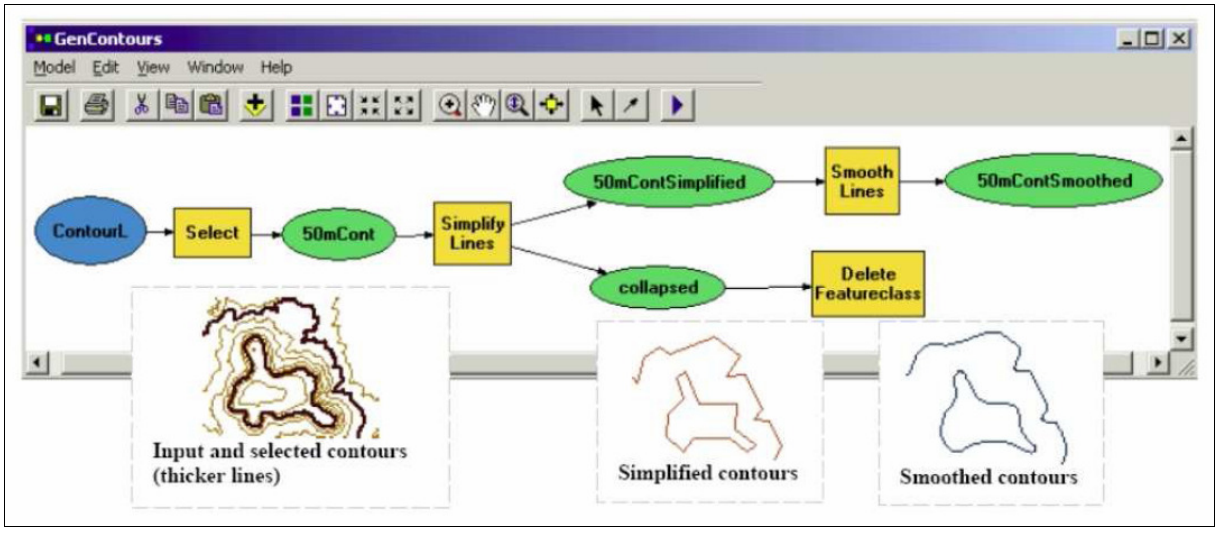


Figure A.10 Exemple de définition d'un processus de généralisation avec l'outil ModelBuilder (Lee, 2003)

Cette approche a pour avantage d'être simple à mettre en œuvre mais ne permet pas d'obtenir des résultats satisfaisants dès qu'il est nécessaire de tenir compte des caractéristiques implicites des objets ou de leur contexte.

A.III.4 Approches globales

Nous regroupons, dans cette catégorie, toutes les approches visant à résoudre le problème de l'automatisation de manière globale en tenant compte en même temps de l'ensemble des objets à généraliser.

A.III.4.1 Approches par minimisation

Les approches par minimisation proposent de résoudre le problème de l'automatisation de la généralisation par une analyse au niveau des coordonnées des points composant les objets. Les contraintes cartographiques sont traduites sous forme de forces appliquées aux points. L'objectif de la généralisation est de trouver un point d'équilibre entre ces forces. Deux méthodes principales ont été proposées pour résoudre ce problème.

La première consiste à modéliser chaque contrainte sous la forme d'une équation dont les inconnues sont les coordonnées finales des points composant les géométries des objets. Les équations dépendent des coordonnées initiales ainsi que des forces appliquées sur les différents points. L'ensemble de ces équations forme un système d'équations surcontraint qui est résolu par une méthode de minimisation telle que la méthode des moindres carrés (Sester, 2000 ; Harrie, 2001 ; Harrie et Sarjakosji, 2002 ; Sester, 2005) ou la méthode des éléments finis (Burghardt & Meier, 1997 ; Højholt, 2000 ; Bader et al., 2005). La figure A.11 donne un exemple de résultat obtenu par la méthode des moindres carrés.

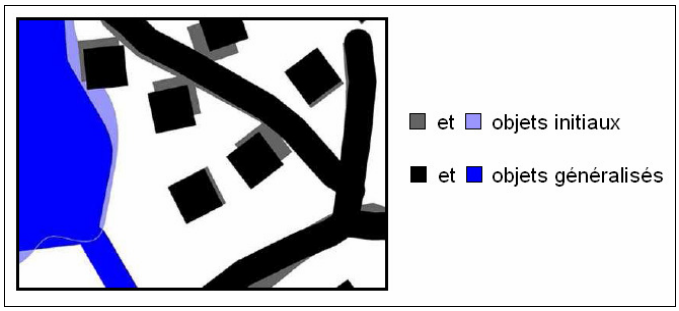


Figure A.11 Exemple de généralisation par moindres carrés. D'après (Harrie, 2001)

La seconde méthode consiste à calculer, à chaque itération, l'ensemble des forces et à déduire pour chaque point, le micro-déplacement résultant de ces forces (Fritsch, 1997 ; Baeijs, 1998). Le processus est réitéré jusqu'à ce qu'un point d'équilibre soit atteint. Les équations ne portent plus directement sur les coordonnées finales souhaitées, mais sur les évolutions élémentaires que subissent les différents points.

Ces approches par minimisation permettent d'obtenir de très bons résultats dans de nombreux cas. L'un de leurs principaux défauts est qu'elles ne permettent pas de recourir à des opérations brusques, telles que l'élimination d'objets, qui sont pourtant essentielles pour les grands changements d'échelle. Un autre problème vient de la prise en compte de l'ensemble des objets durant le processus de généralisation, car ces approches nécessitent dans la plupart des cas des calculs lourds. Il est donc important de bien partitionner l'espace de façon à limiter la quantité d'objets considérés en même temps. Ce partitionnement peut avoir une grande influence sur la qualité des résultats (Lemarié, 2003).

A.III.4.2 Approches par optimisation globale

Nous regroupons sous ce terme *approches par optimisation globale* les approches basées sur la recherche d'un maximum d'une fonction globale d'utilité. A chaque itération, une opération de généralisation est appliquée, et l'état résultant est validé ou non en fonction de la fonction de coût.

Une première approche proposée par (Ware & Jones, 1998 ; Ware et al., 2003) consiste à utiliser le recuit simulé (Kirkpatrick, 1983) pour la recherche d'un maximum. A chaque itération, sont tirés au hasard un objet, un algorithme de généralisation et des valeurs pour les paramètres de l'algorithme. Si l'évolution de la fonction de coût est satisfaisante, le nouvel état est validé. Le recuit simulé est une métaheuristique permettant de faire varier le taux accepté de détérioration de la fonction de coût pour la validation des états : au début du processus, des détériorations importantes de la fonction de coût sont acceptées alors qu'elles ne le sont plus vers la fin du processus. Le processus peut ainsi éviter d'être piégé dans un maximum local tout en garantissant sa convergence. La figure A.12 donne un exemple de résultat obtenu par cette approche.

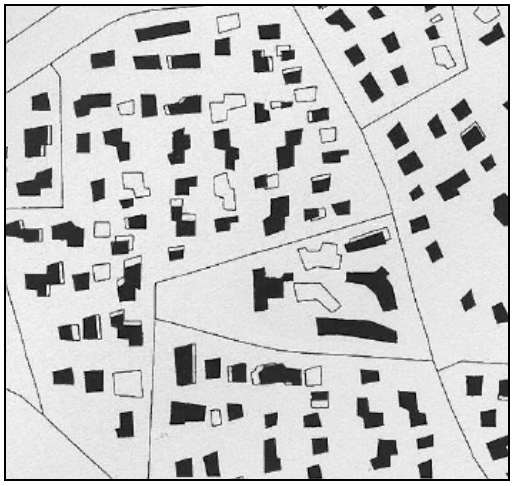


Figure A.12 Exemple de généralisation par recuit simulé. D'après (Ware et al., 2003)

Une seconde approche, proposée par (Wilson et al., 2003), consiste à utiliser un algorithme génétique (Holland, 1975) pour la recherche d'un maximum de la fonction de coût. Le principe des algorithmes génétiques est de faire subir à une population d'individus (qui est généralement tirée au hasard au départ) différentes opérations d'évolution (combinaisons entre individus et mutations) ainsi que des sélections qui vont favoriser les individus les plus prometteurs vis-à-vis de la fonction de coût. Les individus représentent ici l'ensemble des coordonnées, traduites en mode binaire, des vecteurs de déplacement de tous les points des objets. La figure A.13 donne un exemple de résultat obtenu par cette approche.

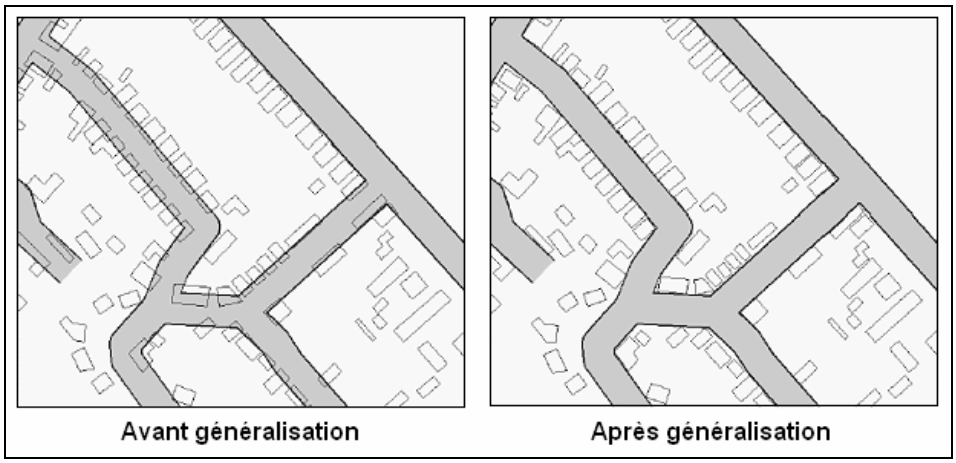


Figure A.13 Exemple de généralisation par algorithme génétique. D'après (Wilson et al., 2003)

La principale difficulté de ce type d'approche provient de la complexité à définir une fonction de coût globale pertinente. Il faut, en effet, que cette fonction de coût tienne compte de l'ensemble des contraintes cartographiques et puisse résumer en un nombre la qualité générale d'un état. On retrouve aussi les mêmes problèmes que pour les approches par minimisation : le calcul de la fonction de coût nécessite de prendre simultanément en considération un grand nombre d'objets, ce qui peut s'avérer extrêmement coûteux en temps de calcul. Il n'est donc généralement pas possible de traiter l'ensemble des objets d'une carte en même temps. Une approche pour résoudre ce problème consiste à partitionner l'espace géographique en zones pouvant être traitées indépendamment. Cette approche pose le problème du choix d'un partitionnement pertinent.

A.III.5 Approches locales

A.III.5.1 Principes

Les approches globales sont basées sur une analyse globale de la situation. Au contraire, les approches locales sont basées sur un découpage du problème général en une multitude de sous-problèmes qui sont résolus indépendamment (Brassel & Weibel, 1988 ; Müller & Wang, 1992 ; Plazanet, 1996 ; Ruas & Plazanet, 1996 ; Mustière, 1998).

Les approches locales nécessitent donc de disposer d'outils et de méthodes permettant le découpage du problème en sous-problèmes ainsi que d'outils permettant la résolution de ces sous-problèmes. Il s'agit donc de disposer de méthodes visant à déterminer quels sont les objets ou les parties d'objets sur lesquels lancer le processus de généralisation et de déterminer ensuite quels traitements appliquer sur ces derniers (Shea & McMaster, 1989 ; McMaster & Shea, 1998).

Le modèle proposé par (Ruas & Plazanet, 1996) s'intéresse particulièrement à la question du découpage de l'espace de travail. Le modèle de (Weibel & Dutton, 1999), qui est complémentaire au premier modèle et qui traite du problème du choix des traitements, reprend l'idée de (Beard, 1991) de traduire les spécifications de l'utilisateur sous forme de contraintes explicites. Ces dernières interviennent dans l'évaluation des situations ainsi que dans le choix des algorithmes à appliquer.

Ce type d'approche a été utilisé dans de nombreux travaux. Parmi ceux-ci, GALBE (Mustière, 1998 ; Mustière, 2005) qui se propose de généraliser des tronçons routiers en découpant puis en appliquant sur chaque partie des tronçons différents algorithmes en fonction des conflits d'empâtement (situation où le symbole se superpose à lui-même). La figure A.14 donne un exemple de résultats obtenus avec GALBE.

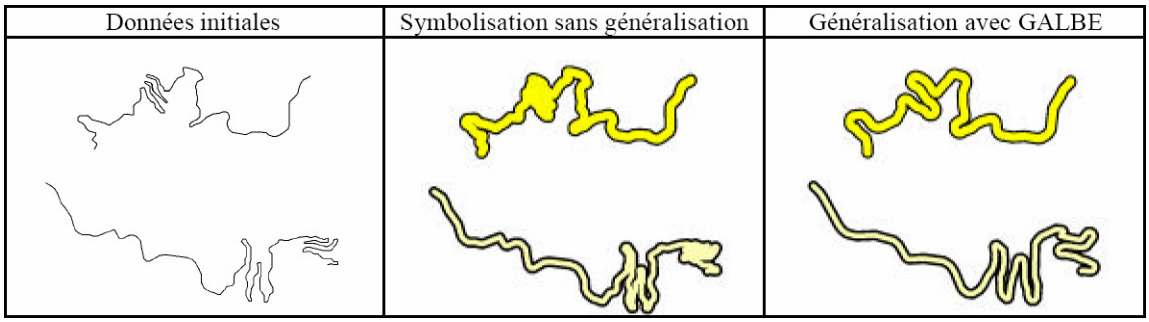


Figure A.14 Exemple de résultats obtenus avec GALBE. D'après (Mustière, 2001)

A.III.5.2 Approches basées sur le paradigme agent

D'autres travaux, également basés sur l'utilisation explicite de contraintes, ont proposé d'utiliser le paradigme agent pour modéliser les objets géographiques (Ruas, 1999 ; Duchêne, 2004 ; Jabeur, 2006 ; Ruas & Duchêne, 2007).

Le paradigme agent est issu de l'intelligence artificielle. Il n'existe pas de définition universellement acceptée du concept d'agent. On trouve ainsi de nombreuses définitions dans la littérature (Shoham, 1993 ; Ferber, 1997 ; Weiss, 1999 ; Chaib-Draa et al., 2001 ; Russel & Norvig, 2006).

L'une des plus citées est celle de (Ferber, 1997) qui définit un agent par « une entité physique ou virtuelle :

a. qui est capable d'agir dans un environnement,
b. qui peut communiquer directement avec d'autres agents,
c. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
d. qui possède des ressources propres,
e. qui est capable de percevoir (mais de manière limitée) son environnement,
f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
g. qui possède des compétences et offre des services,
h. qui peut éventuellement se reproduire,
i. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit. »

On peut identifier deux tendances principales dans les définitions du concept d'agent : certaines s'intéressent à la définition d'un agent pris isolément, alors que d'autres, à l'image de la définition de (Ferber, 1997), intègrent dans le concept d'agent, son appartenance à une société d'agents (un système multi-agent).

Les travaux sur l'automatisation de la généralisation, qui se basent sur le paradigme agent, se situent dans cette deuxième tendance. Chaque objet géographique de la base de données à généraliser (bâtiments, routes, etc.) est modélisé sous la forme d'un agent. L'ensemble de ces agents constitue une communauté dans laquelle chaque agent a pour tâche de maximiser une fonction locale d'évaluation en s'appliquant diverses opérations de généralisation.

Le modèle AGENT (Ruas, 1999 ; Lamy et al., 1999 ; Barrault et al., 2001) est basé sur ce type d'approche et propose une organisation de type hiérarchique entre les agents. Ainsi, un agent *ville* est en contact avec les agents *groupement de bâtiments* qui le composent, qui sont eux-mêmes en contact avec les agents *bâtiment* qui les composent. Ce type d'organisation hiérarchique est particulièrement bien adapté pour les zones où des relations hiérarchiques se dégagent naturellement telles que les zones urbaines. La figure A.15 donne des exemples de résultats obtenus avec le modèle de généralisation AGENT. Nous présentons plus en détail ce modèle, qui sert de base applicative à ce travail de thèse, dans la partie suivante (A.IV).

Figure A.15 Exemples de résultats obtenus avec le modèle de généralisation AGENT

Le modèle CartACom (Duchêne, 2004) est également basé sur une approche agent, mais propose une communication directe entre les agents sans structure hiérarchique. Le choix des opérations de généralisation à appliquer résulte d'une concertation entre les agents impliqués dans un même conflit cartographique. Ce modèle est particulièrement bien adapté aux zones de type rural où il est difficile de définir des relations hiérarchiques. La figure A.16 donne un exemple de résultat obtenu avec CartACom.

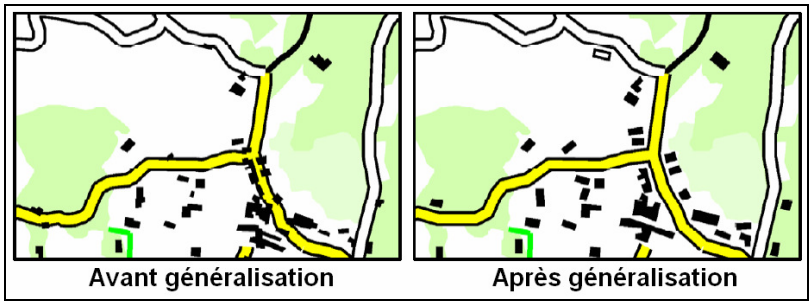


Figure A.16 Exemple de résultats obtenus avec CartACom. D'après (Duchêne, 2004)

Il existe également d'autres modèles appartenant aux approches locales et basés sur l'utilisation du paradigme agent dont certains proposent un autre niveau d'analyse. C'est le cas du modèle GAEL (Gaffuri, 2008) destiné à généraliser les thèmes « champ ». Ces thèmes, qui ne sont en général pas traités, ou le sont mal, par la plupart des modèles de généralisation automatique, concernent les classes de données géographiques qui ont une valeur en tout point de l'espace. Des exemples de thèmes champ sont l'occupation du sol ou le relief. Dans le modèle GAEL, les agents ne représentent plus des objets géographiques, mais les points composant les objets. On passe d'un niveau micro à un niveau sub-micro. La généralisation consiste alors, pour ces agents points, à trouver, par déplacements successifs, une valeur d'équilibre qui permette au mieux de satisfaire toutes les contraintes portant sur eux. A noter que, contrairement aux approches globales par minimisation, l'analyse et les transformations ont ici un cadre local : on ne cherche pas à traiter l'ensemble des données en même temps. La figure A.17 donne un exemple de résultats obtenus pour la déformation du relief dans le cadre de la préservation de l'altitude de bâtiments.

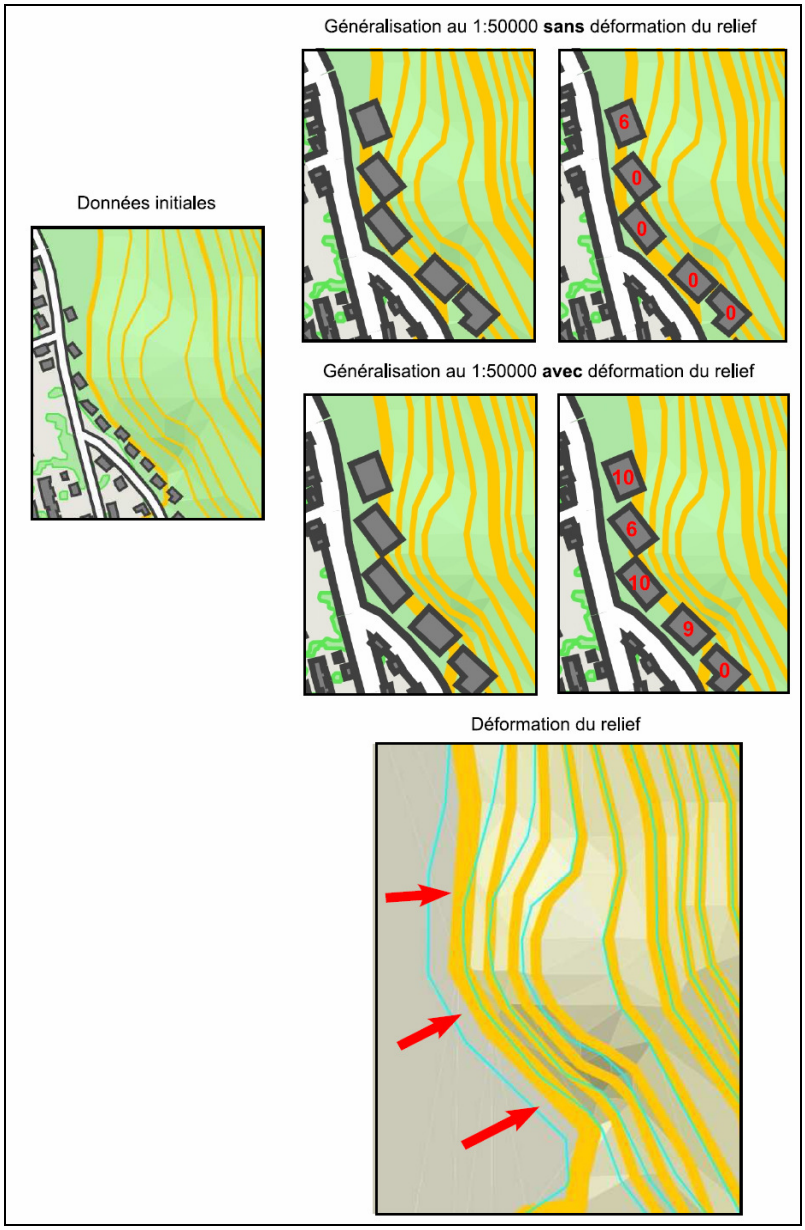


Figure A.17 Exemple de résultats obtenus avec GAEL. D'après (Gaffuri, 2008)

A.IV Le modèle AGENT : un modèle de généralisation automatique fonctionnant par exploration informée d'arbres d'états

A.IV.1 Principe général

Le modèle AGENT, qui fait partie des approches locales d'automatisation de la généralisation, a été initié par (Ruas, 1999). Il a été ensuite enrichi et réimplémenté dans le cadre du projet AGENT (Lamy et al., 1999 ; Barrault et al. 2001 ; Projet AGENT, 2001) puis a fait l'objet d'une réimplémentation dans le cadre du logiciel Radius Clarity™ (1Spatial, 2007). D'autres travaux ont également contribué à l'enrichissement de ce modèle (Regnauld, 2001 ; Duchêne, 2004). Il est actuellement utilisé à l'IGN pour la production de diverses séries de cartes.

Le modèle AGENT se situe dans la continuité des travaux de (Beard, 1991) : les spécifications souhaitées pour la carte finale sont traduites sous forme de contraintes. Ces dernières interviennent dans l'évaluation des situations ainsi que dans le choix des algorithmes à appliquer.

Comme évoqué dans la partie précédente, le modèle AGENT est basé sur le paradigme agent. Il propose de modéliser les objets géographiques (routes, bâtiments, etc.) sous forme d'agents géographiques, faisant d'eux les entités décisionnelles du processus de généralisation. La figure A.18 donne des exemples d'agents géographiques : un bâtiment ou une route peuvent ainsi être des agents. Un groupe de bâtiments ou même une ville peuvent également être modélisés sous forme d'agents. Les agents géographiques ont pour tâche de gérer leur propre généralisation. Ils disposent pour cela de capacités d'auto-analyse leur permettant de choisir au mieux les algorithmes de généralisation à s'appliquer en vue de satisfaire les contraintes cartographiques portant sur eux-mêmes.

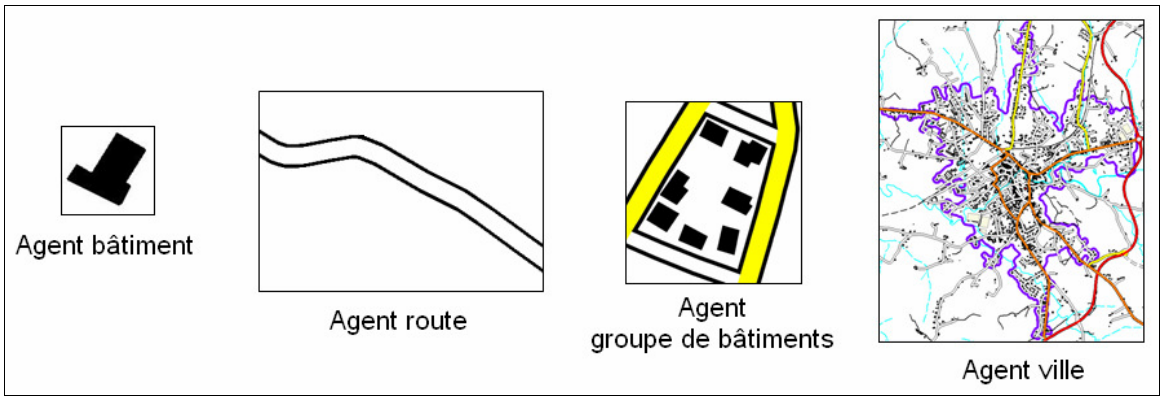


Figure A.18 Exemple d'agents géographiques du modèle AGENT

La généralisation nécessite souvent de prendre en compte le contexte spatial dans lequel se trouvent les objets. En effet, certains conflits cartographiques ainsi que certaines opérations de généralisation ne concernent pas un unique objet géographique, mais un ensemble d'objets.

Par exemple, l'algorithme de déplacements de bâtiments dans un îlot présenté figure A.9, page 12, nécessite de prendre en considération l'ensemble des bâtiments et des routes de l'îlot.

Le modèle AGENT propose, pour faire face à ce problème, d'avoir recours à deux niveaux d'analyse : le niveau micro, qui correspond aux objets géographiques pris de façon unitaire, et le niveau meso, qui désigne les groupes d'objets géographiques. Un agent meso peut aussi bien être composé d'agents micro que d'agents meso. Ces niveaux d'analyse permettent d'obtenir une structure multi-niveaux hiérarchique de type arbre (au sens informatique du terme) dont les feuilles sont des agents micro. La figure A.19 donne un exemple de hiérarchie possible pour l'espace urbain.

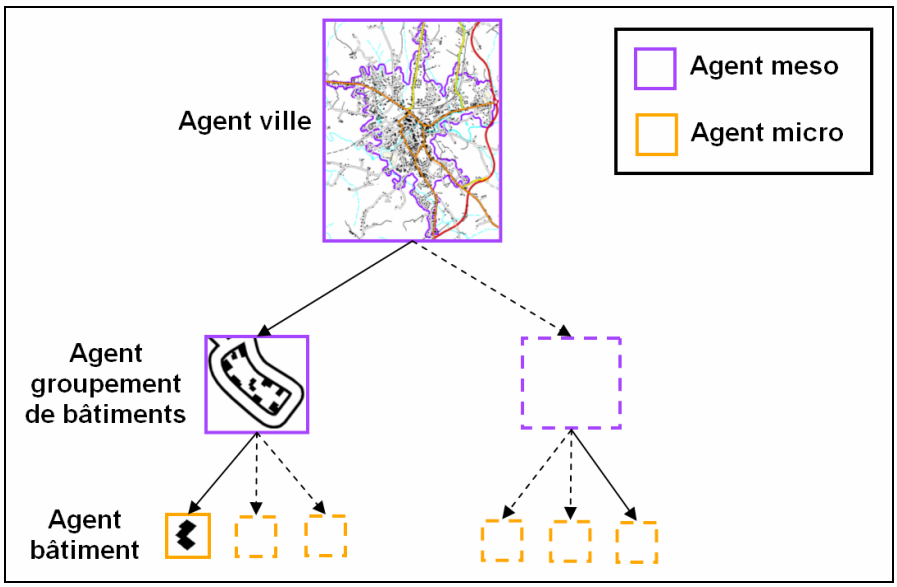


Figure A.19 Exemple de hiérarchie d'agents géographiques

Les agents meso ont pour rôle de gérer les conflits cartographiques et les opérations qui impliquent plusieurs objets du groupe qu'ils représentent. Par exemple, un agent meso *groupement de bâtiments*, doit, entre autres, régler les conflits de superpositions entre routes et bâtiments et peut appliquer une opération de déplacement des bâtiments. Les agents meso ont également pour rôle de superviser la généralisation des sous-agents qui les composent. Ainsi, un agent meso *groupement de bâtiments* peut demander aux agents *bâtiment* le composant de se généraliser individuellement.

A.IV.2 Les contraintes

Comme nous l'avons évoqué dans la partie précédente, le modèle AGENT suit la proposition de (Beard, 1991) en exprimant explicitement les contraintes cartographiques. Les contraintes du modèle AGENT traduisent les spécifications du produit cartographique souhaité. Elles ont pour rôle de guider la généralisation des agents géographiques. Chaque contrainte relative à un agent est modélisée sous la forme d'un objet, au sens informatique du terme, lié à cet agent. La figure A.20 donne le diagramme de classes des agents géographiques et de leurs contraintes.

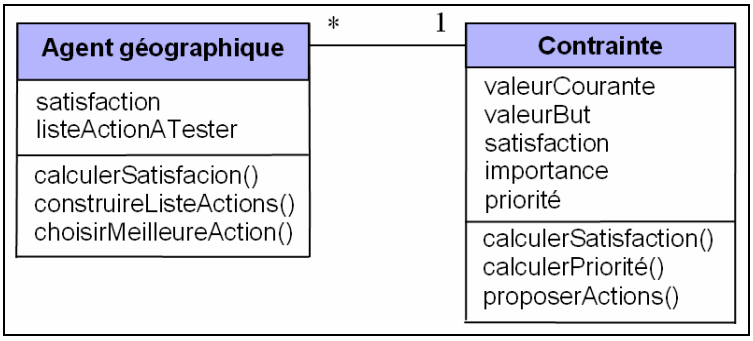


Figure A.20 Diagramme de classes des agents géographiques et des contraintes dans le modèle AGENT

Un objet "contrainte" assure, pour un agent géographique donné, le suivi du degré de violation de la contrainte. Il a aussi pour rôle de calculer pour son agent associé, en fonction de sa satisfaction et éventuellement d'autres mesures, des listes d'actions (des algorithmes de généralisation) à tester. Par exemple, si la contrainte de taille d'un agent *bâtiment* n'est pas satisfaite (le bâtiment est trop petit), elle peut lui proposer de s'appliquer une action de grossissement.

Chaque contrainte porte les attributs suivants :

1 **Valeur courante** : elle traduit, pour un état de l'agent géographique, les caractéristiques courantes de la contrainte.

2 **Valeur but** : elle traduit les spécifications du produit souhaité et représente ce que devrait être idéalement la valeur courante.

3 **Satisfaction** : elle reflète le degré de satisfaction d'une contrainte. C'est une interprétation de l'écart entre la valeur courante et la valeur but. Elle est représentée par un entier compris entre 1 et 10 (10 = contrainte parfaitement satisfaite).

4 **Importance** : elle correspond à l'importance, pour la qualité du résultat final, qu'une contrainte soit satisfaite. Elle est représentée par un entier compris entre 1 et 5 (1 = pas important ; 5 = très important).

5 **Priorité** : elle reflète l'urgence de traitement d'une contrainte. Plus une contrainte possède une priorité élevée, plus les actions susceptibles de l'améliorer seront prioritaires. Elle est calculée pour chaque état, en fonction de la satisfaction. Elle est représentée par un entier compris entre 1 et 5 (1 = pas urgent ; 5 = très urgent).

La figure A.21 donne un exemple de jeu de contraintes définies pour un agent *bâtiment*. Certaines contraintes peuvent proposer des actions à leur agent associé comme la contrainte *de taille* qui peut proposer à l'agent *bâtiment* de grossir, la contrainte *de granularité* qui peut lui proposer de simplifier sa forme ou la contrainte *d'équarrité* qui peut lui proposer de rendre droits certains de ses angles. D'autres contraintes, au contraire, n'ont qu'un rôle passif comme la contrainte *de conservation de la forme*. Leur rôle consiste, alors, juste à refuser certains états dans lesquels elles sont très insatisfaites.

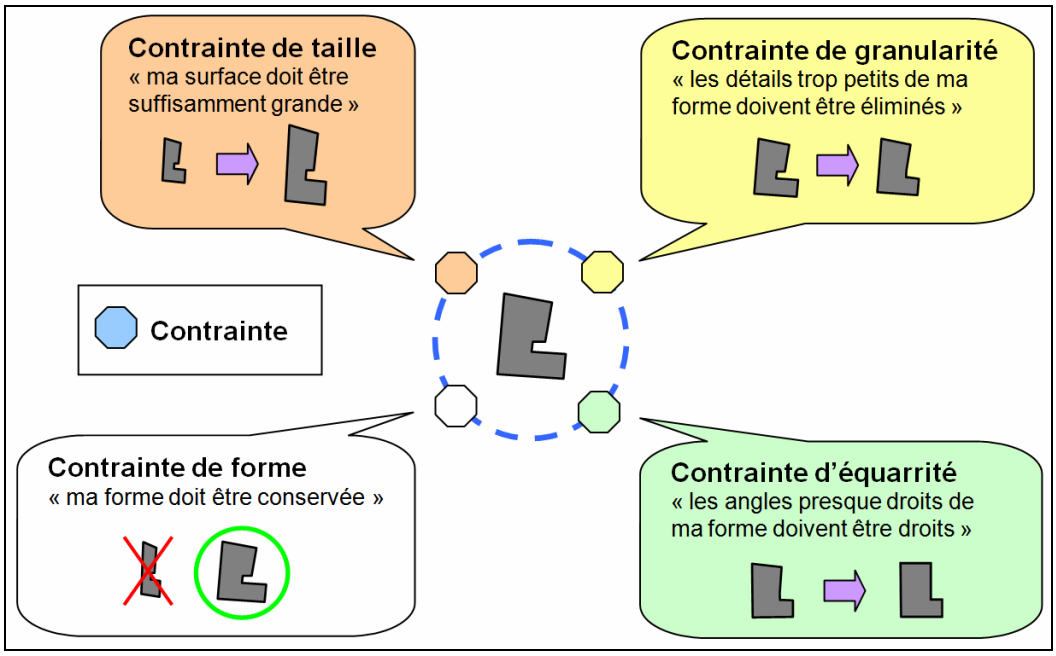


Figure A.21 Exemple de contraintes définies pour un agent *bâtiment*

A.IV.3 Cycle d'actions

Pour satisfaire au mieux ses contraintes, un agent géographique réalise, quand il est activé, un cycle d'actions. Durant ce cycle d'actions, l'agent géographique teste les différentes actions proposées par ses contraintes afin d'atteindre un état parfait (où toutes ses contraintes sont parfaitement satisfaites), ou à défaut un état aussi satisfaisant que possible (figure A.22).

Son fonctionnement est le suivant :
L'agent commence par caractériser son état initial. Pour cela, ses contraintes calculent leur satisfaction, puis l'agent calcule à son tour, à partir de celles-ci, sa propre satisfaction. Une fois caractérisé, l'agent construit une liste d'actions à s'appliquer. Celle-ci est composée des actions proposées par ses contraintes.
L'agent ordonne ensuite cette liste en fonction de trois critères : le premier est la priorité de la contrainte qui a proposé l'action. Plus une contrainte est prioritaire, plus les actions qu'elle propose sont testées en priorité. Si plusieurs contraintes ont la même priorité, l'agent teste en premier les actions de la contrainte la moins satisfaite. Enfin, si une contrainte propose plusieurs actions, celles-ci sont ordonnées en fonction de leur poids. En effet, chaque action proposée se voit attribuer un poids. Plus une action a un poids élevé, plus elle aura de chances d'être testée en priorité.
Si la liste d'actions est vide, l'agent, s'il n'est pas dans son état initial, retourne à son état précédent, et vérifie s'il lui reste des actions à tester pour cet état. Tant que l'agent ne trouve pas d'action à tester et qu'il n'est pas dans son état initial, il continue à retourner à des états précédents. Si l'agent se trouve dans son état initial et que sa liste d'actions est vide, il retourne au meilleur état qu'il a trouvé durant le cycle d'actions et se désactive.
Dans le cas où la liste d'actions n'est pas vide, l'agent applique l'action la plus prioritaire et la supprime de la liste. L'agent se caractérise ensuite dans ce nouvel état. Si cet état est un état parfait, l'agent se désactive, dans le cas contraire, l'agent teste sa validité. Si l'état est valide, l'agent revient à l'étape de construction d'une liste d'actions. S'il ne l'est pas, l'agent retourne à son état précédent et vérifie s'il lui reste des actions à tester pour cet état.

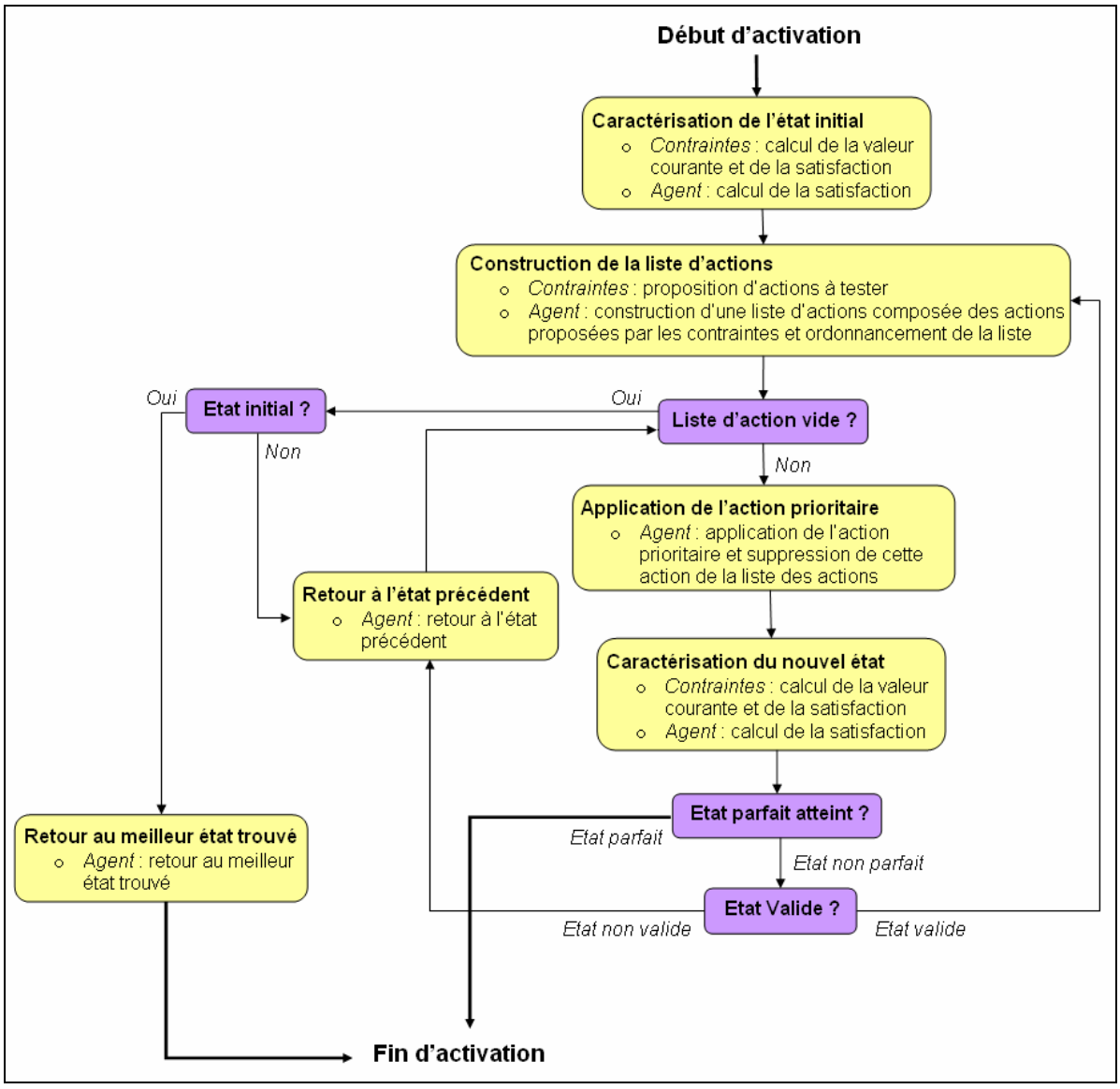


Figure A.22 Cycle d'actions d'un agent dans le modèle AGENT

Le cycle d'actions se traduit par un parcours en profondeur de type « meilleur d'abord » d'un arbre d'états. La figure A.23 donne un exemple d'arbre obtenu pour la généralisation au 1 : 50 000 d'un bâtiment.

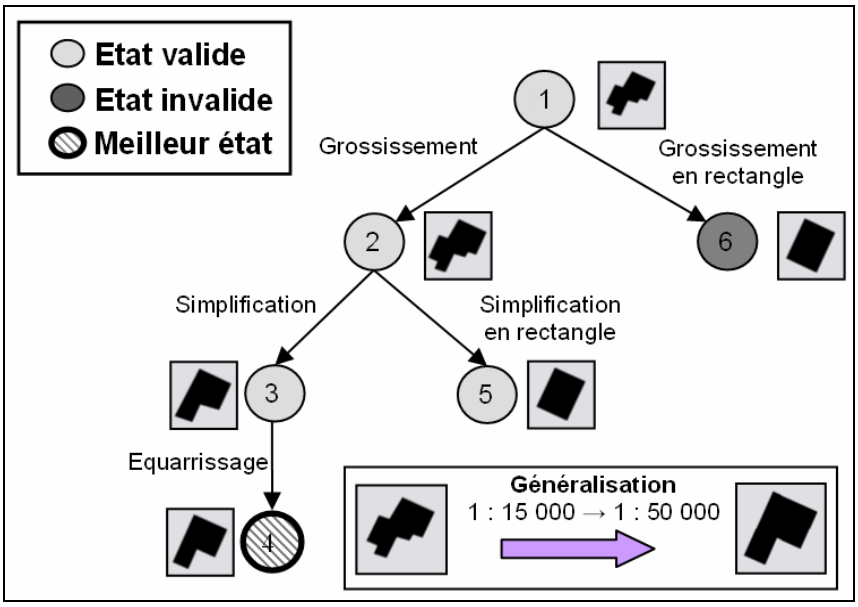


Figure A.23 Exemple d'arbre d'états obtenu pour la généralisation d'un bâtiment

Le passage d'un état à un autre correspond à l'application par l'agent de l'une des actions proposées par l'une au moins de ses contraintes. Pour chaque nouvel état, l'agent calcule sa satisfaction globale. Cette satisfaction globale définit la qualité de l'état. Elle est égale à la moyenne des satisfactions des contraintes de l'agent pondérée par leur importance. Par exemple si un agent a deux contraintes C_1 et C_2 ayant respectivement pour importance et pour satisfaction, 3 et 8 pour C_1, et 5 et 2 pour C_2, la satisfaction globale de l'agent pour cet état est de 4.86.

Pour un état donné, l'ordre d'application des actions dépend en premier lieu de la priorité des contraintes. Si deux contraintes ont la même priorité, les actions testées en priorité seront celles proposées par la contrainte la moins satisfaite. Lorsqu'une contrainte propose plusieurs actions, l'ordre d'application des actions dépend du poids de chaque action. Plus une action a un poids élevé, plus elle est testée en priorité.

Une action peut être utilisée à plusieurs reprises par séquence d'actions. Il est fréquent que la séquence d'actions menant au meilleur état nécessite l'application à différentes étapes du processus de généralisation, de la même action (avec éventuellement des paramètres différents). Le nombre de séquences possibles d'actions est donc infini. Par ailleurs, il n'est pas assuré qu'il existe, pour chaque généralisation, un état parfait.

Afin d'assurer la convergence du processus de généralisation, le modèle AGENT fait intervenir un critère de validité, permettant de stopper l'exploration d'une branche qui semble a priori non prometteuse. L'introduction de ce critère de validité a pour conséquence de ne plus garantir l'optimalité du meilleur état trouvé compte tenu des actions disponibles.
Etant donné que les algorithmes de généralisation nécessitent des calculs assez lourds et que le nombre d'objets géographiques à traiter peut se révéler très élevé, le critère de validité choisi sera généralement tel qu'il entraîne un élagage assez sévère permettant ainsi de limiter le nombre d'états parcourus.

A.IV.4 Connaissances procédurales en jeu

Nous avons décrit, dans les parties précédentes, le fonctionnement du modèle AGENT. Celui-ci fait intervenir un certain nombre de connaissances procédurales, c'est-à-dire de connaissances ayant trait à la façon dont les arbres d'états sont parcourus. Ces connaissances occupent une place très importante dans ce modèle.

Le modèle AGENT comprend deux grands groupes de connaissances :

- Connaissances communes à tous les agents géographiques : le critère de fin de cycle et le critère de validité des états.
- Connaissances liées à une classe de contraintes : connaissances relatives à la priorité des contraintes et relatives au choix des actions à proposer (et de leur poids).

Nous notons $Satisfaction_{Ctr}(e)$, la valeur de la satisfaction de la contrainte *ctr* pour un état *e* donné. Nous notons de la même manière $Valeur_m(e)$ la valeur de la mesure *m* pour un état *e* donné.

Critère de fin de cycle

Le cycle d'actions s'arrête (et l'agent se désactive) lorsque l'agent géographique n'a plus d'action à tester pour son état initial ou lorsqu'un état parfait est trouvé.
Un état parfait désigne un état pour lequel toutes les contraintes de l'agent sont parfaitement satisfaites (i.e. ont une satisfaction de 10).

Critère de validité

Différents critères de validité ont été proposés pour ce modèle dans la littérature (Regnauld, 2001 ; Taillandier, 2007a). Nous présentons ici celui implémenté dans la version 2.6 du SIG *Radius Clarity*™.
Un état est valide si :

- la satisfaction de la contrainte qui a proposé l'action ayant mené à ce nouvel état a augmenté.
- ET s'il y a *amélioration partielle* par rapport à chacun des états déjà parcourus. Une *amélioration partielle* signifie qu'au moins l'une des contraintes a vu sa satisfaction progresser. Soit un agent géographique avec deux contraintes C_1 et C_2 et un nouvel état ayant respectivement pour valeur de satisfactions pour ces contraintes 4 et 7. Si pour l'un des états déjà parcourus, ces mêmes contraintes ont pour valeur de satisfaction 5 et 7, le nouvel état ne sera pas valide. En effet, aucune des deux satisfactions de l'état courant n'est supérieure à celles correspondantes de l'état déjà visité. Par contre, il y a *amélioration partielle* par rapport à un état où C_1 et C_2 ont respectivement pour valeurs de satisfaction 6 et 6, car C_2 a vu sa satisfaction progresser.

La deuxième condition de ce critère de validité permet d'assurer la convergence du cycle d'actions. En effet, la satisfaction des contraintes étant un entier compris entre 1 et 10, le nombre d'états différents en termes de satisfaction des contraintes est donc fini (il est égal à $10^{\text{nb de contraintes}}$). Or le second critère entraîne l'invalidité d'un nouvel état si celui-ci est similaire à un état déjà visité en termes de satisfaction des contraintes.

Connaissances de priorité des contraintes

Chaque contrainte dispose d'une base de règles définissant, en fonction de sa satisfaction, sa priorité. La base de règles couvre l'ensemble des valeurs possibles de la satisfaction (comprise entre 1 et 10).

Les règles sont du type :

Si ($Satisfaction_{Contrainte}$(état Courant) > 5)
Alors la priorité de la contrainte = 3

Connaissances d'application des actions

Chaque contrainte dispose d'une base de règles définissant, pour un état donné de l'agent géographique (caractérisé par la satisfaction de la contrainte et par éventuellement d'autres mesures), quelles actions doivent être appliquées et avec quel poids.

Les règles sont du type :

Si ($Satisfaction_{Contrainte}$(état Courant) > 5 et $Valeur_m$(état courant) < 3
Alors proposer l'action Act_1 avec un poids = p_1, et l'action Act_2 avec un poids = p_2 avec (p_1, p_2) entiers ∈ [1,5][2]

Définir un jeu de connaissances procédurales pour une classe d'agents consiste donc à définir, pour chaque contrainte, comment évolue sa priorité en fonction de sa satisfaction et quelles actions proposer en fonction de sa satisfaction et éventuellement d'autres mesures. La figure A.24 donne un exemple de jeu de connaissances défini pour les agents *bâtiments*.

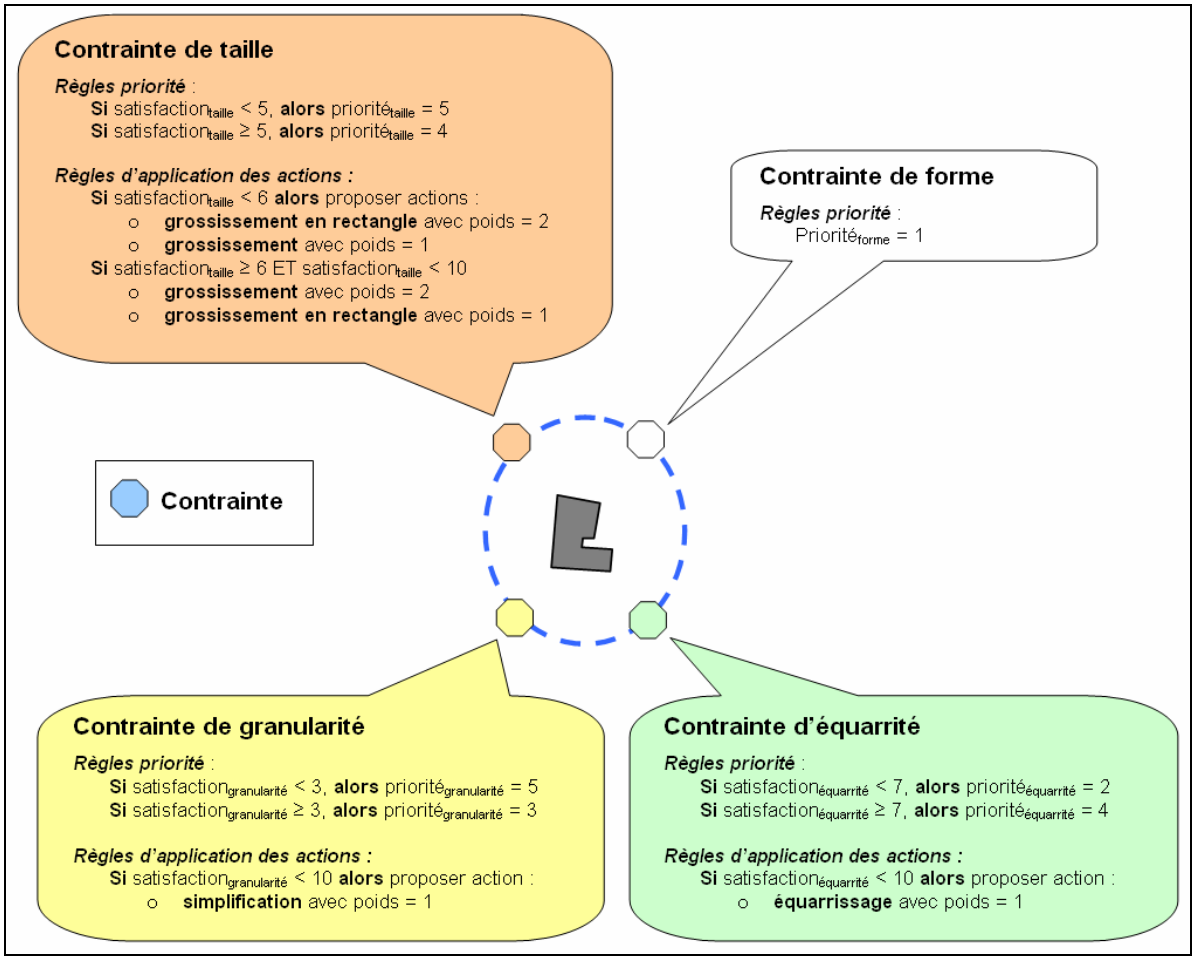


Figure A.24 Exemple de jeu de connaissances pour les agents *bâtiment*

A.V Enjeu de la révision des connaissances procédurales dans les systèmes de généralisation fonctionnant par exploration informée d'arbres d'états

A.V.1 Amélioration de l'efficience et de l'efficacité

La qualité des résultats de généralisation obtenus avec un système de généralisation basé sur l'utilisation de connaissances procédurales telles que celles du modèle AGENT, est directement liée à la qualité des connaissances. Ainsi, des connaissances mauvaises peuvent entraîner des problèmes d'efficience et/ou d'efficacité. L'efficacité d'un système désigne sa capacité à obtenir un résultat de qualité. L'efficience désigne sa capacité à trouver ce résultat en utilisant le moins de ressources possible (en termes de temps processeur).

Un mauvais guidage de la part des connaissances peut entraîner des problèmes d'efficacité si l'élagage utilisé pour l'exploration de l'arbre est sévère, ou des problèmes d'efficience si l'élagage utilisé est plus faible.

Disposer de bonnes connaissances procédurales dans le cadre de la généralisation de données géographiques est d'autant plus important que le nombre d'objets à généraliser est souvent très important.

Pour illustrer ces propos, prenons l'exemple de la ville d'Orthez (Pyrénées-Atlantiques) qui compte 10 000 habitants. Cette dernière est composée d'environ 280 groupements de bâtiments. Si l'on considère que le temps moyen pour l'application d'une action et pour la caractérisation du nouvel état est de l'ordre de la seconde (temps réaliste compte tenu des expérimentations), et si le nombre moyen d'états parcourus est de 50, le temps moyen de généralisation par groupement de bâtiments sera de l'ordre de 50 secondes et le temps nécessaire à la généralisation de l'ensemble des groupements de bâtiments sera donc de l'ordre de 4 heures. Ce temps est très important sachant qu'Orthez n'est que la 839^{ième} ville de France en termes de population et que de nombreux objets, appartenant à d'autres classes que celle des groupes de bâtiments, doivent être aussi généralisés.

Il est donc nécessaire de limiter au maximum le nombre d'états parcourus, mais ceci en conservant une qualité de résultat acceptable. En effet, s'il est toujours possible de faire « corriger » la généralisation de quelques objets par un opérateur humain, il faut que ce nombre d'objets soit suffisamment faible pour que le temps passé à effectuer les corrections ne devienne pas prohibitif.

Définir à la main un jeu de connaissances procédurales pour une classe d'objets peut se révéler très ardu et fastidieux. Il est généralement nécessaire de passer de très longues heures à tester les différentes actions avant de pouvoir obtenir des connaissances procédurales convenables, et cela même pour des spécialistes de la généralisation et du modèle AGENT.

A.V.2 Evolutivité du système

Un autre problème lié aux connaissances concerne l'évolutivité du système. En effet, ces dernières sont valables pour un état donné du système mais nécessitent d'être réadaptées pour tout changement d'état du système.

Les changements d'état du système peuvent provenir de nombreuses sources. Une première peut être l'ajout de nouveaux éléments tels que de nouvelles actions ou de nouvelles mesures. Il devient alors nécessaire, si l'on souhaite prendre en considération, dans le processus de généralisation, ces nouveaux éléments, de modifier en conséquence les connaissances procédurales.

Une autre source de changement d'état du système peut être une simple modification des besoins, et donc de la fonction d'évaluation des états. Cette modification peut nécessiter une révision en profondeur des connaissances. Typiquement, si l'on dispose de connaissances qui privilégient la satisfaction d'une contrainte par rapport à une autre et que, pour une nouvelle utilisation du système de généralisation, il est désormais nécessaire de disposer de connaissances privilégiant plutôt la satisfaction de la deuxième contrainte par rapport à la première, les connaissances de priorité des contraintes et d'application des actions devront être réadaptées.

Ces deux types de changement d'état du système nécessitent donc une réadaptation des connaissances. Or, ces changements sont suffisamment fréquents pour qu'il soit important de disposer d'outils permettant de réadapter simplement les connaissances procédurales sans que cela ne demande de longues heures de travail de la part des opérateurs humains. Il devient donc très intéressant de disposer d'un processus totalement automatique de révision des connaissances qui puisse être appliqué sans qu'aucune expertise humaine n'ait à intervenir.

A.VI Objectif de la thèse

Nous avons présenté dans ce chapitre notre contexte applicatif : celui de la généralisation de données géographiques. Nous avons en particulier introduit les concepts de données géographiques, de généralisation de données géographiques et de généralisation cartographique. Nous avons ensuite présenté un état de l'art des approches d'automatisation de la généralisation avant de revenir plus en détail sur le modèle AGENT qui est basé sur l'exploration informée d'arbres d'états. Nous avons introduit dans une dernière partie les enjeux liés à la révision des connaissances dans des systèmes de généralisation basés sur ce type de modèle et nous avons en particulier montré l'utilité que pouvait avoir un processus automatique de révision des connaissances.

L'objectif applicatif de cette thèse est de proposer un modèle de révision automatique des connaissances procédurales contenues dans les systèmes de généralisation automatique basés sur le modèle AGENT ou sur des modèles du même type.

Si la révision automatique des connaissances procédurales a un intérêt dans le cadre des systèmes de généralisation de données géographiques, elle en a également pour la plupart des systèmes basés sur l'exploration informée d'arbres d'états. En effet, les difficultés, liées à la définition de connaissances pertinentes et à leur évolution, sont des problèmes récurrents dès que la définition des connaissances procédurales nécessite une expertise du domaine. On se retrouve alors face au problème classique en intelligence artificielle du *goulot d'étranglement* de l'acquisition des connaissances (Feigenbaum, 1977).

Notre objectif applicatif peut donc être généralisé à tout système fonctionnant par exploration informée d'arbres d'états. Nous posons donc comme objectif scientifique pour cette thèse de proposer un modèle de révision automatique des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états.

Nous présentons dans le chapitre suivant les problématiques liées à la révision des connaissances contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états ainsi qu'une vue d'ensemble de notre modèle et de notre approche de révision automatique des connaissances procédurales.

Chapitre B
Problématique et approche générale

B.I Introduction

Nous avons posé dans le chapitre précédent le contexte applicatif de cette thèse : celui de la généralisation de données géographiques. Nous avons ainsi introduit la notion de généralisation de données géographiques et présenté un état de l'art des méthodes d'automatisation de ce processus. Cet état de l'art s'est conclu par la présentation d'un modèle de généralisation automatique de données géographiques fonctionnant par exploration informée d'arbres d'états, le modèle AGENT. L'analyse de ce dernier a permis de mettre en évidence les principales difficultés posées par ce type de modèle : la définition des connaissances permettant de guider l'exploration des arbres d'états ainsi que leur évolution. Nous avons généralisé ce constat à tout système fonctionnant par exploration informée d'arbres d'états et ainsi introduit l'objectif de cette thèse qui consiste à proposer un modèle de révision automatique des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états.

Nous présenterons dans ce second chapitre les problématiques liées à la révision automatique des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états ainsi qu'une vue d'ensemble de notre approche générale de révision des connaissances procédurales.

Nous commencerons pour cela par introduire un état de l'art de l'acquisition et de la révision des connaissances en intelligence artificielle puis de leur application dans le cadre de la généralisation automatique de données géographiques. Cette première partie permettra d'introduire de nombreux outils que nous utiliserons dans ce travail de thèse et de présenter divers travaux touchant directement l'objectif que nous nous sommes fixé.

Une fois ce contexte introduit, nous proposerons une première formalisation du problème de la révision par l'expérience des connaissances contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états. Cette formalisation permettra dans une troisième partie d'introduire les problématiques liées à ce problème.

Nous présenterons dans une dernière partie notre approche générale ainsi que notre modèle général. En vue de remplir notre objectif applicatif, nous présenterons comment appliquer ces derniers au cadre du modèle AGENT. Nous proposerons, à cet effet, des enrichissements apportés au modèle AGENT afin de permettre la mise en œuvre de notre approche.

B.II Acquisition et révision automatique de connaissances

B.II.1 Acquisition et révision des connaissances en intelligence artificielle

B.II.1.1 Introduction

Ce travail de thèse porte sur la révision des connaissances. Cette dernière pose le problème de l'intégration de nouvelles informations dans des bases de connaissances déjà définies. Le premier système capable d'une telle révision est l'Homme. En effet, nous sommes tous très régulièrement confrontés au problème de la correction de nos connaissances en raison de nouvelles informations perçues dans le monde réel.

En informatique, ce problème se révèle capital dans de nombreux domaines tels que ceux de la gestion de bases de données et de l'intelligence artificielle où de nombreuses applications utilisent des bases de connaissances.

Parmi ces applications se trouvent les systèmes experts qui firent partie des premiers travaux en intelligence artificielle (Buchanan et al., 1969) et qui connurent un grand succès dans les années 80. Un système expert est un programme qui cherche à reproduire, pour un domaine particulier, les mécanismes cognitifs d'un expert. Il procède pour cela par raisonnements déductifs à partir de sa base de connaissances et des faits observés. Un système expert est constitué de trois composants : un moteur d'inférence, une base de connaissances et une base de faits. La plupart des systèmes experts s'appuient sur des formalismes logiques (logique des propositions ou logique des prédicats).

Les premiers systèmes experts, bien qu'utilisés à leur époque avec succès dans de nombreux domaines, souffraient de nombreuses limitations. L'une d'elles concernait la construction de la base de connaissances. Eward Feigenbaum, l'un des pères du premier système expert DENDRAL, a formulé en 1977 le problème maintenant classique en intelligence artificielle du « goulot d'étranglement de l'acquisition des connaissances » (Feigenbaum, 1977). Il est en effet souvent délicat et fastidieux de formaliser les connaissances des experts dans un formalisme utilisable par une machine. Une approche pour faire face à ce goulot d'étranglement est d'utiliser les techniques de l'apprentissage artificiel. Ces dernières permettent de construire des connaissances à partir de données et peuvent donc aider à la construction de bases de connaissances. Nous présentons en partie B.II.1.2 les principes de l'apprentissage artificiel et nous détaillons en partie B.II.1.3 le cas de l'apprentissage inductif supervisé qui permet de construire des connaissances à partir d'exemples étiquetés par un expert et qui nous intéressera particulièrement dans le cadre de cette thèse.

Pouvoir acquérir de nouvelles connaissances est indispensable dans le cadre d'un processus de révision des connaissances, mais cela n'est pas suffisant. En effet, réviser des connaissances implique non seulement de construire de nouvelles connaissances mais aussi de prendre en considération les connaissances existantes. Des travaux en intelligence artificielle ont déjà porté sur ces points. Nous les présentons en partie B.II.1.5.

B.II.1.2 Généralités sur l'apprentissage artificiel

(Cornuéjols et al., 2002, p.vi) proposent de définir l'apprentissage artificiel par :

> « Cette notion englobe toute méthode permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle ».

L'objectif de l'apprentissage artificiel est de construire automatiquement des connaissances. Il permet ainsi de doter des programmes de capacités d'amélioration et d'adaptation par l'expérience. Il se base sur l'analyse de données afin d'extraire un modèle de la réalité. Ces données peuvent soit provenir de l'observation d'une source externe, soit être le résultat d'une introspection (figure B.1).

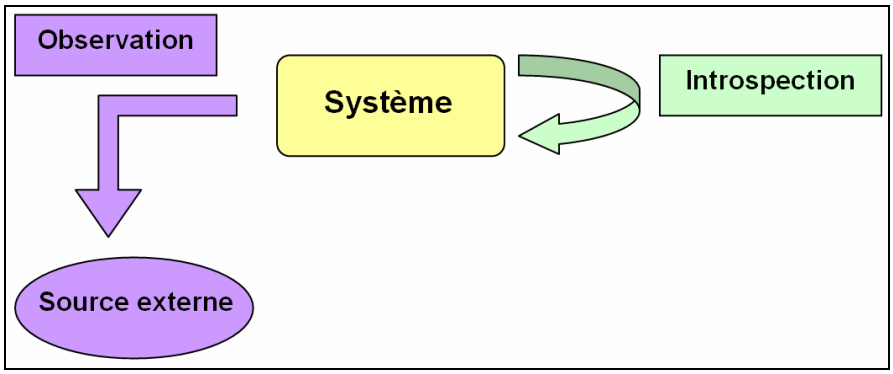


Figure B.1 Apprentissage observation/réflexion

On distingue classiquement trois types d'apprentissage artificiel :

- **Apprentissage supervisé** : le cadre classique de l'apprentissage supervisé est celui d'un apprenant qui doit apprendre, à partir de données étiquetées par un expert (ou oracle), à bien étiqueter lui-même l'ensemble des données possibles. L'apprenant doit, à cet effet, trouver ou approximer la fonction (appelée aussi hypothèse) qui permet d'affecter la bonne étiquette à ces données. L'apprentissage supervisé est basé sur un principe inductif permettant de passer d'observations particulières à une loi générale.
- **Apprentissage non supervisé** : l'apprentissage non supervisé se place dans un cadre où aucun oracle n'est utilisé. L'apprenant doit trouver par lui-même, dans un ensemble de données, les régularités sous-jacentes.
- **Apprentissage par renforcement** : l'objectif de l'apprentissage par renforcement est d'apprendre un comportement à partir d'une séquence de perceptions, d'actions, et de récompenses. Les actions que l'apprenant effectue sur l'environnement produisent des valeurs de retour qui guident l'apprenant dans sa tâche d'apprentissage.

Ce travail de thèse s'intéresse à la révision des connaissances à partir de l'expérience. L'expérience que nous utilisons dans le cadre du processus de révision résulte de l'analyse des traces d'exécution et donc d'une introspection. Le système, au travers du module de révision, est capable d'analyser les traces et possède sa propre expertise. Nous sommes, de ce fait, dans le cadre d'un apprentissage supervisé dans lequel le système joue lui-même le rôle de l'expert. Nous nous intéresserons donc, dans la suite de cette présentation des principes de l'apprentissage artificiel, à l'apprentissage supervisé.

B.II.1.3 L'apprentissage supervisé

L'objectif de l'apprentissage supervisé est d'apprendre automatiquement, à partir d'un ensemble restreint d'observations étiquetées par un oracle, une fonction (appelée hypothèse) permettant de reproduire ou d'approximer l'étiquetage de l'oracle (figure B.2). Le protocole d'apprentissage est défini par trois composantes :

- **Les acteurs** : trois acteurs interviennent dans le processus d'apprentissage : l'environnement, l'oracle (ou professeur) et l'apprenant.
  L'environnement est supposé stationnaire. Il fournit des données (objets ou observations) à l'oracle et à l'apprenant. L'oracle retourne une étiquette pour chaque objet qu'on lui présente. L'apprenant doit trouver dans l'espace des hypothèses $\mathcal{H}$, celle étant la plus adaptée en vue de proposer des étiquettes similaires à celles fournies par l'oracle. Les couples (x_i,u_i), qui associent une étiquette u_i à une observation x_i, sont appelés exemples.
- **La tâche d'apprentissage** : elle consiste en la recherche par l'apprenant de l'hypothèse qui approxime le mieux les réponses retournées par l'oracle. Lorsque cette réponse est un booléen (ou qu'il n'y a que deux classes possibles), on parle d'apprentissage de concept. S'il s'agit d'une réponse à valeur discrète, on parle de classification et si la réponse est à valeur continue, on parle de régression.
- **Un principe inductif** : il permet de passer d'observations particulières à une loi générale.

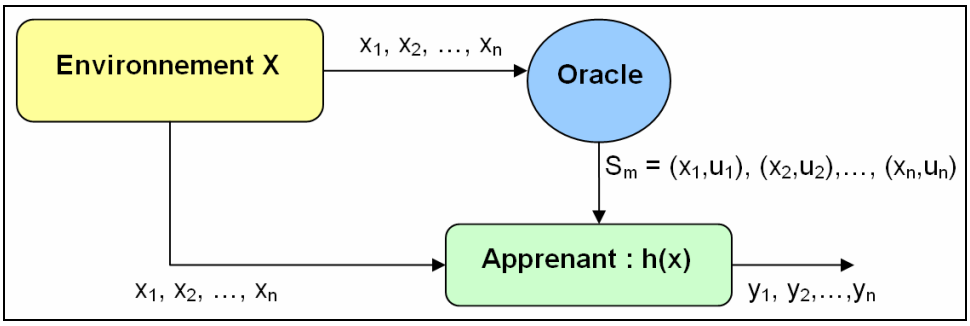


Figure B.2 Apprentissage supervisé

Notion de biais

Une notion importante de l'apprentissage supervisé est celle de biais. Les biais désignent toutes les restrictions formulées par l'apprenant afin que celui-ci puisse déduire une hypothèse à partir des données étiquetées fournies. Il est indispensable de disposer de biais pour induire une loi générale à partir d'observations. Sans eux l'apprenant ne pourra jamais attribuer d'étiquettes à des données jamais rencontrées auparavant (Mitchell 1980).

On appelle biais de représentation le choix du langage utilisé pour décrire les hypothèses et les données. Ces choix sont déterminants pour la réussite de l'apprentissage. Ainsi, si le langage de représentation des hypothèses est trop restreint, il sera alors impossible d'approximer correctement la fonction cible. Au contraire, s'il est trop large, la meilleure solution sera très difficile à trouver.

Diverses sources d'erreurs peuvent intervenir dans l'apprentissage supervisé. Une première provient du biais inductif (parfois simplement appelé biais) qui représente l'inadéquation entre l'espace des hypothèses et celui auquel appartient la fonction cible. Ce type d'erreur est appelé erreur d'approximation. Une seconde source d'erreurs provient du fait que l'hypothèse

proposée est apprise à partir d'un échantillon de données (appelé jeu d'apprentissage) et non à partir de l'ensemble des données possibles. Elle peut donc dépendre du jeu d'apprentissage et de ses particularités. Cette erreur d'estimation est souvent appelée variance. Plus le langage de représentation des hypothèses est large, plus la variance est en général importante. On appelle surapprentissage ou *overfitting* le cas où l'hypothèse est trop dépendante des données du jeu d'apprentissage et où celle-ci n'est que le résultat d'un apprentissage par cœur de ces dernières et n'est plus représentative de la fonction recherchée. Ce phénomène peut être aggravé par la présence de bruit dans les données d'apprentissage. Ce bruit peut venir d'erreurs de classification de la part de l'oracle (ce qui est très fréquent dans le cas de données réelles) mais également d'un manque d'expressivité du langage de représentation des données.

La réduction du biais (et donc de l'erreur d'approximation) nécessite la complexification de l'espace des hypothèses mais cette complexification peut entraîner dans un même temps une hausse de la variance. Il faut donc trouver un compromis entre le biais et la variance. La figure B.3 illustre ce problème : l'objectif de l'apprentissage est ici d'apprendre la valeur de y en fonction de la valeur de x. Il s'agit donc d'une régression. A gauche, la fonction a été apprise avec un biais trop fort qui ne permet d'apprendre que des hypothèses ayant la forme de droites et ne permet donc pas d'apprendre de fonctions représentatives de la fonction recherchée. A droite, l'espace des hypothèses a été complexifié permettant ainsi d'apprendre des hypothèses plus riches. Malheureusement, cette complexification a entraîné une hausse de la variance : la fonction apprise est trop proche des exemples du jeu d'apprentissage (surapprentissage) et en particulier de leur imperfection.

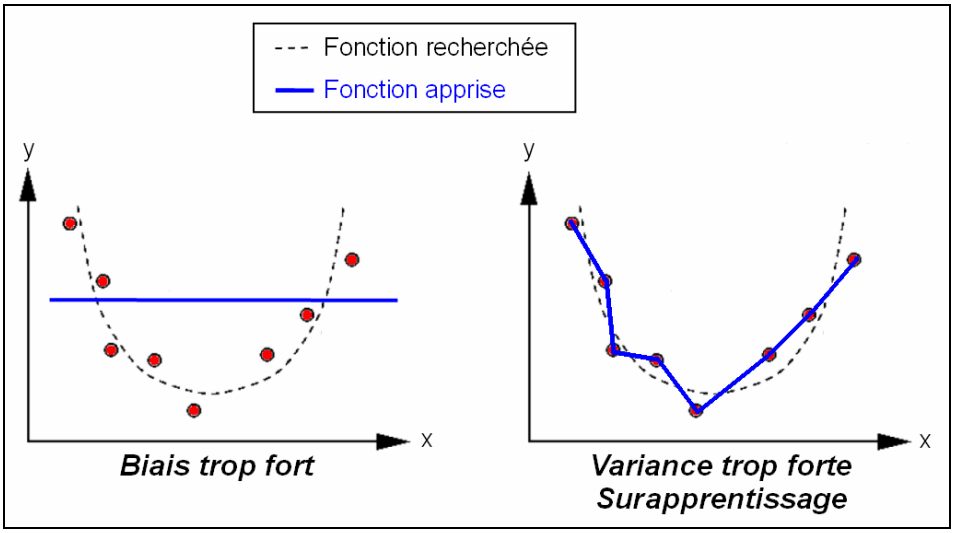


Figure B.3 Compromis biais/variance

L'apprentissage supervisé en tant que problème de minimisation d'une fonction d'erreur

L'objectif de l'apprentissage supervisé est de trouver dans l'espace des hypothèses $\mathcal{H}$, celle qui est la plus adaptée au problème posé et qui permet de fournir les réponses les plus proches de celles de l'oracle. Il s'agit donc pour l'apprenant de chercher l'hypothèse qui minimise l'écart entre les sorties qu'elle propose et celles fournies par l'oracle. De façon à déterminer cet écart, on mesure, pour chaque couple (x_i, u_i), une perte que l'on appelle aussi coût $l(u_i,h(x_i))$ qui évalue le coût d'avoir affecté à une observation x_i, l'étiquette $y_i = h(x_i)$ au lieu de l'étiquette désirée u_i.

Nous notons X, l'ensemble des observations possibles et U, l'ensemble des étiquettes possibles. Le risque fonctionnel aussi appelé risque réel s'exprime de la façon suivante :

$$R_{Réel}(h) = \int_{Z=X\times U} l(u_i, h(x_i))dP(x,u)$$

P(x,u) désigne la loi de probabilité jointe sur X × Y. Par exemple, dans le cas de variables aléatoires discrètes, elle désigne la probabilité d'observer le couple (x,u). Plus une hypothèse fournit des sorties éloignées de celles proposées par l'oracle pour les exemples fréquemment rencontrés, plus son erreur réelle est élevée.

Le problème de l'apprentissage supervisé consiste donc à trouver une hypothèse minimisant ce risque à partir d'un échantillon d'apprentissage.

Il est impossible en pratique de calculer le risque réel car ce calcul nécessite de connaître l'ensemble des exemples ainsi que la fonction P(x,u). De nombreux principes inductifs se basant sur une approximation de ce risque réel à partir d'un échantillon d'exemples (le jeu d'apprentissage) ont donc été proposés. Parmi ceux-ci, le principe inductif ERM est le plus employé. Il se base sur la minimisation du risque empirique R_{emp} qui représente le risque calculé sur le jeu d'apprentissage.

$$R_{emp}(h) = \frac{1}{n}\sum_{i=1}^{n} l(u_i, h(x_i))$$

Cet estimateur est biaisé par le choix du jeu d'apprentissage. Plus celui-ci est représentatif de l'ensemble des exemples possibles, plus l'estimateur est proche de l'erreur réelle.

L'augmentation de la complexité de l'espace des hypothèses peut permettre de diminuer l'erreur empirique, mais elle peut aussi entraîner des problèmes de surapprentissage et donc l'augmentation de l'erreur réelle. Plus l'espace des hypothèses est riche, plus il y a de risques de surapprentissage et donc plus il est important de disposer d'un jeu d'apprentissage réellement représentatif.

Des travaux sur la théorie de l'apprentissage (Valiant, 1984 ; Blumer et al., 1989 ; Vapnik, 1995) tels que ceux réalisés dans le cadre de l'analyse PAC (Angluin, 1988), ont permis de donner un cadre plus formel à la relation qui existe entre l'erreur empirique et l'erreur réelle.

Evaluation de l'apprentissage supervisé

Un problème classique de l'apprentissage artificiel et plus particulièrement de l'apprentissage supervisé est celui de son évaluation. Il n'est, en effet, pas possible en pratique de calculer l'erreur réelle. De plus, le taux d'erreur empirique calculé sur l'ensemble d'apprentissage est peu fiable car biaisé (il y a un risque de surapprentissage).

Il existe néanmoins différentes méthodes pour évaluer la qualité d'une hypothèse apprise par apprentissage. La plus simple consiste à se servir, en plus du jeu d'apprentissage, d'un jeu de test totalement indépendant du jeu d'apprentissage. On divise pour cela et de façon aléatoire, le jeu de données initial en deux parties. La première est utilisée comme jeu d'apprentissage et la seconde comme jeu de test. L'apprentissage de l'hypothèse est effectué sur le jeu

d'apprentissage et son évaluation sur le jeu de test. On utilise généralement 2/3 des exemples pour l'apprentissage et le reste pour l'évaluation.

Cette méthode est fiable si l'on dispose d'assez d'exemples. En effet, il faut prendre en considération le fait qu'une partie des exemples n'est pas utilisée pour l'apprentissage, ce qui peut nuire à la qualité de l'apprentissage. De plus, il faut disposer d'assez d'exemples pour que l'évaluation effectuée sur le jeu de test ait un sens. Effectivement, si ce dernier n'est absolument pas représentatif de l'ensemble des exemples possibles, l'erreur calculée sur ce jeu sera non représentative de l'erreur réelle.

Lorsque peu d'exemples sont disponibles, une méthode classique d'évaluation est la validation croisée (Stone, 1974). Son principe est le suivant :

- On sépare l'ensemble des exemples à disposition E, en k parties $\{P_i\}_{(1\leq i\leq k)}$.
- Pour chaque partie P_i, on apprend une hypothèse h_i sur l'ensemble $\{E - P_i\}$ puis on l'évalue sur P_i.
- Le taux d'erreur estimé total est égal à la moyenne des taux d'erreur évalués sur les P_i.

On utilise généralement 10 parties. La validation croisée offre l'avantage de fournir une évaluation non biaisée mais elle nécessite d'utiliser l'algorithme d'apprentissage k fois, ce qui peut s'avérer assez lourd en termes de calcul lorsque l'on dispose d'une base d'exemples très importante. Cette technique ne donne pas qu'une hypothèse en sortie mais k hypothèses. Obtenir des hypothèses très différentes est généralement le signe d'une variance trop importante et donc d'une mauvaise qualité de l'apprentissage. La meilleure stratégie pour le choix de l'hypothèse finale est de refaire un apprentissage sur l'ensemble d'apprentissage E complet et de garder l'hypothèse obtenue.

B.II.1.4 Caractérisation d'un problème d'apprentissage

Poser un problème d'apprentissage requiert d'identifier et de choisir des paramètres propres à la tâche à réaliser. Une première étape consiste à définir un objectif. Celui-ci peut aussi bien consister en l'acquisition de connaissances (apprentissage de concept, classification, découverte de lois) qu'en l'amélioration des performances d'un système (adaptation, amélioration en ligne).

Une seconde étape consiste à définir le protocole d'apprentissage utilisé et plus particulièrement à choisir la méthode d'apprentissage à employer. Ce choix peut dépendre de nombreux paramètres propres au problème d'apprentissage rencontré. Ainsi, le fait de posséder ou non des exemples étiquetés, de pouvoir interagir directement avec un oracle, que les exemples soient ou non présentés tous à la fois ou au contraire un par un selon un certain tirage, peut avoir une influence majeure sur le choix de la méthode d'apprentissage. Un autre paramètre à prendre en considération est la nature et la qualité des entrées. On peut, en effet, travailler sur des données du type numérique, symbolique (binaire, nominal, séquentiel, logique, …) ou mixte. Leur qualité peut dépendre de leur origine, de leur représentativité ainsi que du bruit qu'elles peuvent comporter. Enfin, un dernier paramètre important concerne le caractère interprétable du modèle prédictif que l'on désire construire (est-il directement compréhensible par un humain ?).

Les algorithmes d'apprentissage sont souvent classés suivant deux tendances. La première est appelée symbolique et est issue de l'intelligence artificielle. Les algorithmes appartenant à cette tendance travaillent sur des symboles et fournissent la plupart du temps des modèles

prédictifs interprétables. Des exemples d'algorithmes de type symbolique sont les algorithmes d'induction d'arbre de décision (Breiman et al., 1984 ; Quinlan, 1993), les algorithmes d'apprentissage par explication (Mitchell et al., 1986) ou encore les algorithmes issus de la programmation logique inductive (Quinlan, 1990 ; Muggleton, 1995). L'autre tendance est l'apprentissage numérique. Elle est issue des statistiques. Les algorithmes appartenant à cette tendance travaillent sur des données numériques. Cette deuxième tendance comprend, entre autres, les algorithmes tels que ceux basés sur l'apprentissage bayésien (John & Langley, 1995 ; Jiang & Zhang, 2006), les réseaux de neurones (Patterson, 1996 ; Shepherd, 1997) et les séparateurs à vaste marge (Vapnik, 1995 ; Platt, 1998). Les modèles prédictifs fournis par ce type d'algorithmes sont généralement ininterprétables.

Des travaux sur la théorie de l'apprentissage ont démontré qu'il n'existait pas d'algorithme d'apprentissage universellement meilleur que les autres. Ainsi le théorème du no-free-lunch (Wolpert & Macready, 1995) affirme que tous les algorithmes se valent. Il démontre, en effet, que quelles que soient les performances d'un modèle prédictif sur un jeu d'apprentissage, celui-ci n'a pas de raison d'être performant pour d'autres jeux d'exemples. Un algorithme d'apprentissage est toujours biaisé pour une certaine classe de problèmes. L'induction ne fait que traduire l'information incluse dans l'échantillon et dans le biais d'apprentissage. Elle ne crée pas d'information. Elle est donc totalement dépendante de la qualité de l'information a priori.

B.II.1.5 Révision des connaissances en intelligence artificielle

Nous avons évoqué jusqu'à présent le cas de l'apprentissage supervisé. Les travaux de ce domaine cherchent à produire des connaissances à partir d'observations particulières. De nombreux algorithmes ont ainsi été proposés. Parmi ceux-ci, très peu proposent d'intégrer des connaissances initiales dans leur processus inductif.

Certains travaux proposent toutefois d'utiliser des connaissances déjà existantes du domaine d'étude afin de faciliter l'apprentissage. Ces connaissances, appelées théorie du domaine, permettent d'apprendre à partir de très peu d'exemples, grâce à des raisonnements qui enrichissent l'expérience. L'une des techniques les plus utilisées dans ce champ de recherche est l'apprentissage par explication appelé aussi EBL (Mitchell et al., 1986). Cette technique peut être utilisée dans le cadre de problèmes de classification comme dans le cadre de problèmes de planification. Son principe est d'utiliser la théorie du domaine pour construire une *explication*. Un système basé sur l'EBL cherche ainsi, pour une instance donnée d'un problème, une explication à la solution obtenue. Cette explication permet de généraliser la solution (ou une solution voisine) à une région de l'espace des instances du problème. Ce type d'algorithme d'apprentissage a principalement été utilisé dans le cadre de l'optimisation de processus de résolution de problèmes. Le système LEX2 (Mitchell et al., 1986) utilise ainsi l'EBL pour apprendre, pour chaque opérateur, ses conditions d'application. D'autres systèmes tels que SOAR (Laird et al., 1986) ou PRODIGY (Minton, 1990) apprennent par EBL des heuristiques de contrôle permettant de guider le système dans le choix des opérateurs à appliquer. Ce type d'approche comporte néanmoins de nombreuses limitations. La première est que la qualité des connaissances apprises dépend très fortement des propriétés de la théorie du domaine. Ainsi, une théorie du domaine incomplète ou incorrecte pose de nombreux problèmes. Une seconde limitation concerne la gestion des données bruitées. En effet, ces approches, basées sur des formalismes logiques, ne gèrent pas ou mal la présence de bruit dans les données d'apprentissage.

Pour revenir à notre problématique de révision des connaissances par l'expérience, nous ne pourrons pas utiliser directement ces travaux. En effet, nous ne possédons pas de théorie du domaine forte (mais seulement un jeu de connaissances initial de qualité incertaine). De plus, nous nous plaçons dans un cadre où les données d'apprentissage peuvent être bruitées.

D'autres travaux intéressants sont à noter dans le cadre de la révision des connaissances tels que ceux qui s'intéressent à l'affinage de bases de connaissances. Ces derniers ont pour objectif l'amélioration de bases de connaissances définies par des experts humains ou automatiquement. La plupart partent de l'hypothèse que les bases de règles sont déjà bonnes et que seule une légère amélioration est suffisante (Carbonara & Sleeman, 1999).
Certains de ces travaux tels que ceux de (Atzmüller & Baumeister, 2006) cherchent avant tout à servir de support à l'utilisateur dans sa révision des connaissances, alors que d'autres, qui se rapprochent plus de la problématique de cette thèse, permettent de les affiner automatiquement.
Dans ce cadre, certains travaux comme ceux de (Rada, 1985 ; Ginsberg et al., 1988 ; Ma & Wilkins, 1991) proposent d'affiner une base de règles par amélioration de règles existantes et par suppression de celles peu pertinentes. Ils n'offrent toutefois pas la possibilité d'ajouter de nouvelles règles dans la base de connaissances.
D'autres travaux, comme ceux présentés par (Smith et al., 1985), s'intéressent à l'affinage de bases de règles par analyse en ligne d'exemples. Ainsi, à chaque exemple présenté, la base de règles est améliorée. Ce type d'approche ne permet pas de profiter de toutes les informations pouvant être tirées de l'analyse d'un ensemble d'exemples. Pouvoir analyser les exemples dans leur ensemble et non exemple par exemple est crucial pour la grande majorité des techniques d'apprentissage supervisé.
Un défaut fréquent des travaux sur l'affinage de bases de règles concerne leur mauvaise gestion du bruit dans les bases d'exemples. Ainsi de nombreux travaux comme ceux de (Ourston & Mooney 1990) ne permettent pas de faire face à des données bruitées. Un second défaut fréquent concerne la complexité du jeu de connaissances obtenu après affinage. Ainsi la plupart des systèmes d'affinage peuvent entraîner une complexification du jeu de connaissances qui peut nuire à sa lisibilité. Un exemple dans ce cas est le système DLGref2 de (Webb, 1993) pour lequel des expérimentations à partir de règles générées par l'algorithme C4.5 ont montré une multiplication du nombre de règles pouvant aller jusqu'à un facteur 6 sur certaines bases de tests.
Beaucoup de travaux en affinage de bases de connaissances prennent pour cadre l'affinage d'un type particulier de bases de règles (par exemple l'affinage de bases de règles de production). Des travaux tels que ceux de (Boswell & Craw, 1999) ont cherché à proposer un modèle plus générique de représentation des connaissances pouvant être adapté à un grand nombre de types de règles. L'objectif de passer par un tel modèle est de permettre l'utilisation d'un système de raffinement plus générique capable de s'adapter à différents types de règles. Il est tout à fait possible, dans le cadre de notre problématique de révision des connaissances, d'envisager, si les connaissances s'y prêtent, l'utilisation d'un modèle de ce type pour représenter les connaissances. Il est, en effet, possible de formaliser par un tel modèle les connaissances de certains systèmes fonctionnant par exploration informée d'arbres d'états tels que ceux basés sur le modèle AGENT. Nous n'aurons toutefois pas recours à un modèle de ce type dans le cadre de ce travail de thèse. Effectivement, nous souhaitons appuyer notre processus de révision sur les spécificités des bases de règles utilisées par le modèle AGENT.

Pour finir sur l'affinage de bases de connaissances, notons que certains travaux tels que ceux de (Kelbassa, 2003) proposent d'affiner des jeux de règles utilisés dans le cadre du raisonnement à partir de cas (RàPC).
Le RàPC est un paradigme de raisonnement qui trouve ses origines dans les sciences cognitives. Il se base sur l'idée d'utiliser des problèmes déjà résolus pour résoudre de nouveaux problèmes. Le cycle du RàPC se décompose en cinq étapes (figure B.4) :

- Etape *d'élaboration* : cette étape consiste à définir le nouveau cas. Les informations nécessaires à la formulation d'un problème sont collectées et structurées de façon à constituer un nouveau cas : le cas cible.
- Etape de *remémoration* : cette étape consiste à sélectionner dans la base de cas des cas similaires au cas cible. Une mesure de similarité est utilisée pour la sélection des cas.
- Etape *d'adaptation* : cette étape consiste à élaborer une solution pour le cas cible par adaptation des solutions des cas sélectionnés dans la précédente étape.
- Etape de *révision* : l'utilisateur peut durant cette étape refuser ou corriger la solution si celle-ci ne lui convient pas ou si celle-ci se révèle inapte à résoudre le problème. Le système peut alors proposer de nouvelles adaptations afin de fournir une solution satisfaisante.
- Etape de *mémorisation* : cette étape consiste à enrichir la base de cas avec le cas cible résolu. L'enrichissement de la base de cas implique de mettre à jour la base de cas et parfois de la réorganiser.

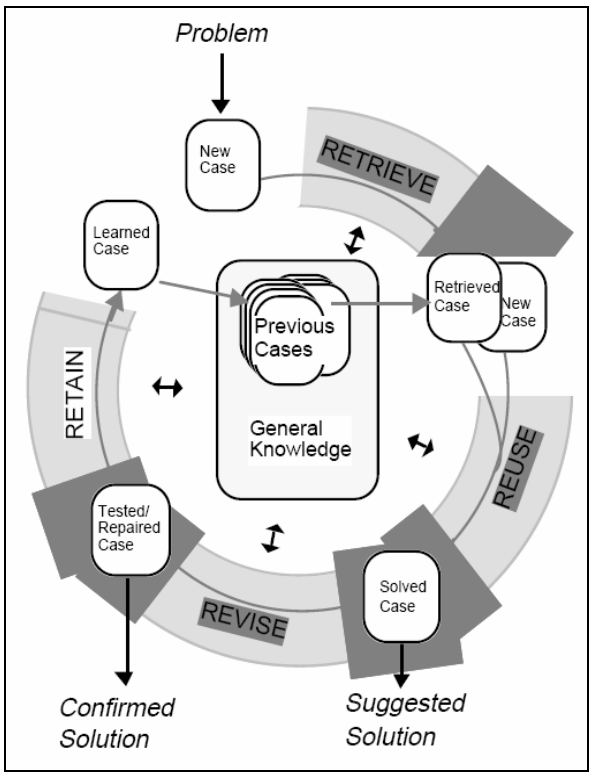


Figure B.4 Cycle du RàPC. D'après (Aamodt & Plaza, 1994)

Le RàPC entre donc dans le cadre des processus de révision des connaissances. En effet, les connaissances (ici la base de cas) sont révisées à chaque nouveau cas résolu. De nombreux travaux portent d'ailleurs sur les problèmes de révision et de réorganisation des connaissances dans les systèmes basés sur ce type de raisonnement (Roth-Berghofer & Iglezakis, 2001).

Le RàPC impose une structure particulière pour les connaissances du système : celle d'un ensemble de cas déjà résolus. Or, dans le cadre de ce travail de thèse, nous souhaitons conserver la structure initiale des connaissances du système et ne pas imposer de structures particulières. Cette contrainte nous garantit, par exemple, de conserver le caractère interprétable des connaissances du modèle AGENT. Il ne nous sera donc pas possible d'utiliser les travaux sur le RàPC directement dans ce travail de thèse.

Nous avons évoqué jusqu'à présent le cas de révisions de connaissances représentées sous forme de bases de règles et de bases de cas. Un autre domaine de recherche très actif concerne la révision de connaissances représentées par des formalismes logiques et plus particulièrement par des formalismes de logiques non monotones.
Les logiques non monotones diffèrent de la logique classique de par leur non respect du principe de monotonie. Ce dernier implique que si *X* et *Y* sont deux ensembles de formules et que si *X* est inclus dans *Y*, alors tout théorème déductible de *X* doit aussi l'être de *Y*.

Le fait de ne pas respecter ce principe de monotonie permet au raisonnement basé sur les logiques non monotone (raisonnement non monotone) de procéder à des inférences sur des informations incertaines ou incomplètes et d'émettre des conclusions pouvant être ensuite rétractées après ajout de nouvelles informations. Un exemple de proposition qui ne peut être exprimée en logique monotone mais peut l'être en logique non monotone est : « généralement, les oiseaux volent ». A moins qu'une autre proposition n'affirme le contraire (par exemple : « un pingouin ne vole pas »), un oiseau est supposé voler.

Le paradigme AGM (Alchourron et al. 1985 ; Gärdernfors, 1988), qui est très utilisé dans le cadre de la révision des connaissances représentées par des formalismes logiques (Konieczny 1999), est basé sur un tel raisonnement. Ce paradigme propose de formaliser trois types d'opérateurs de changement des connaissances : l'opérateur d'expansion, celui de révision et celui de contraction.

- *L'expansion* désigne le passage d'une information indéterminée à une information acceptée ou refusée.
- La *contraction* consiste à passer d'une information acceptée ou refusée à une information indéterminée.
- La *révision* désigne le passage d'une information acceptée à une information refusée ou l'inverse.

Des travaux récents (Ribeiro & Wassermann, 2006) ont proposé d'appliquer le paradigme AGM à la révision d'ontologies basées sur des logiques de description.

Dans le cadre de notre problématique de révision des connaissances, il est possible, pour les connaissances représentées par des formalismes logiques, d'utiliser en partie ces travaux. Néanmoins, s'ils permettent bien de réviser des bases de connaissances en vue de les rendre cohérentes avec de nouvelles informations, ils ne s'intéressent pas à la problématique de la constitution, par l'expérience, de ces nouvelles informations. Or, un point important de ce travail de thèse est de fournir des méthodes de construction automatique d'informations à partir de l'expérience pour les systèmes fonctionnant par exploration informée d'arbres d'états.

B.II.2 Acquisition et révision des connaissances dans le cadre de la généralisation de données géographiques

L'objectif applicatif de ce travail de thèse est de fournir des méthodes de révision des connaissances pour les systèmes de généralisation automatique de données géographiques fonctionnant par exploration informée d'arbres d'états. Comme nous l'avons évoqué dans le cadre du modèle AGENT en partie A.IV, disposer de telles méthodes de révision est très important pour ce type de systèmes.

S'il existe une littérature abondante traitant de la généralisation de données géographiques et des techniques de généralisation employées par les cartographes, traduire ces connaissances dans un formalisme qui puisse être utilisée directement par une machine pose de nombreuses difficultés. On se retrouve en effet face au problème du « goulet d'étranglement de l'acquisition des connaissances » qui a bien été identifié dans le cadre de la généralisation de données géographiques (Rieger & Coulson, 1993; Weibel et al., 1995; Kilpeläinen, 2000).

Comme nous l'avons évoqué dans la partie précédente (B.II.1), l'une des approches permettant d'élargir ce goulot est de faire appel aux techniques de l'apprentissage artificiel. De nombreux travaux ont déjà utilisé ces techniques dans le cadre de l'enrichissement de données géographiques (Plazanet et al., 1998 ; Sester, 1998), de la calibration de système de généralisation (Hubert & Ruas 2003) et de l'automatisation de la généralisation.

Les premiers travaux proposant d'utiliser les techniques de l'apprentissage artificiel pour la généralisation automatique de données géographiques ont été initiés par (Reichenbacher, 1995 ; Weibel et al., 1995). Ces derniers ont ainsi proposé de tracer les actions d'un expert pendant qu'il réalise des généralisations interactives à l'aide du système de généralisation, puis d'apprendre, par apprentissage supervisé, des connaissances à partir de ces traces. Ils ont présenté, dans ce cadre, une première expérimentation dont l'objectif était d'apprendre à définir les valeurs de deux paramètres d'un algorithme de simplification de lignes appliqué à des réseaux routiers. L'expérimentation a permis d'apprendre deux règles qui, bien que très simples, ont démontré la faisabilité de l'approche.

Les travaux de (Mustière, 2001) reprennent cette idée d'utiliser l'apprentissage supervisé pour apprendre des connaissances relatives à la généralisation de données géographiques. L'objectif est ici d'apprendre des connaissances concernant le choix de l'algorithme à employer pour la généralisation des routes. (Mustière, 2001) propose pour cela de décomposer ce problème d'apprentissage en plusieurs sous-problèmes.

Ainsi, au lieu d'apprendre directement à partir d'un jeu de mesures quel algorithme employer, cinq bases de connaissances sont apprises (figure B.5) :

- *Abstraction des mesures utilisées* : cette base de connaissances définit comment passer de mesures quantitatives à des mesures qualitatives. Par exemple, la base de connaissances comprend un jeu de règles permettant de décrire la longueur d'un tronçon routier sous la forme d'une classe « petit », « moyen », ou « grand ».
- *Choix de l'opération* : cette base de connaissances définit, en fonction des mesures qualitatives, si le processus de généralisation doit s'arrêter ou s'il est nécessaire d'appliquer de nouvelles opérations sur les objets concernés. Dans ce deuxième cas, la base de connaissances définit le type d'opération à appliquer (opération de simplification, de focalisation, etc.).

- *Applicabilité des algorithmes* : cette base de connaissances définit pour chaque algorithme si, en fonction des mesures qualitatives et du type d'opération retenu dans la phase précédente, celui-ci peut être appliqué ou non.
- *Choix de l'algorithme* : cette base de connaissances définit, en fonction des mesures qualitatives et de l'applicabilité des algorithmes, l'algorithme qui doit être appliqué.
- *Paramétrage* : cette dernière base de connaissances définit, en fonction des mesures qualitatives et quantitatives, les paramètres de l'algorithme à employer.

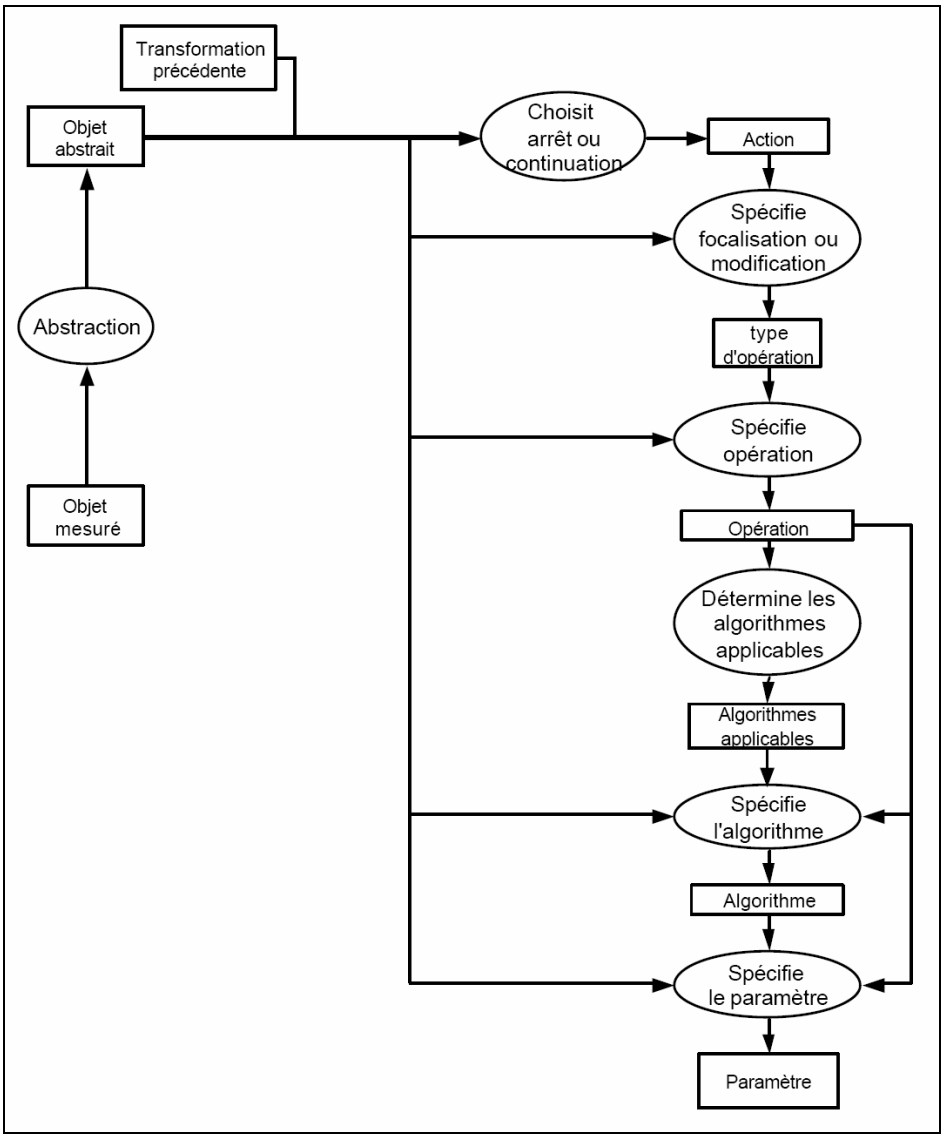


Figure B.5 Moteur du processus généralisation proposé par (Mustière, 2001)

L'intérêt de cette décomposition est d'améliorer la qualité des règles apprises ainsi que leur lisibilité. En effet, la décomposition du problème d'apprentissage permet de réduire pour chaque sous-problème la taille de l'espace des hypothèses, et donc de faciliter l'apprentissage. L'amélioration de la qualité des règles est d'autant plus importante que peu d'exemples ont pu être utilisés pour l'apprentissage. Ce faible nombre s'explique par la fastidiosité du processus d'étiquetage des exemples par des experts en généralisation. Cet étiquetage des exemples par des experts est l'une des principales différences avec notre travail de thèse dans lequel nous cherchons à réviser des connaissances directement à partir de l'expérience, sans avoir recours

à l'intervention d'experts. Il nous est donc possible d'utiliser un nombre d'exemples beaucoup plus important.

Ces différents travaux ont permis de mettre en lumière deux types de difficultés (Ruas et al., 2006). Le premier est celui du choix des connaissances à apprendre ainsi que de leur formalisation (Mustière, 2001 ; Mustière & Ruas, 2004). Le second concerne la collecte des exemples (Mustière, 2001 ; Ruas & Holzapflel, 2003). Le processus d'étiquetage des exemples par des experts se révèle, en effet, souvent fastidieux.

Une approche possible pour faire face au problème de la collecte des exemples est d'intégrer directement dans le système d'information géographique, un environnement facilitant la collecte des exemples ainsi que leur analyse. (Duchêne et al., 2005) propose ainsi l'outil MAACOL intégré au SIG Lamps 2. Cet outil permet d'assister les experts dans le processus d'étiquetage des exemples (figure B.6). Il fournit également des outils d'analyse des connaissances définies par les experts

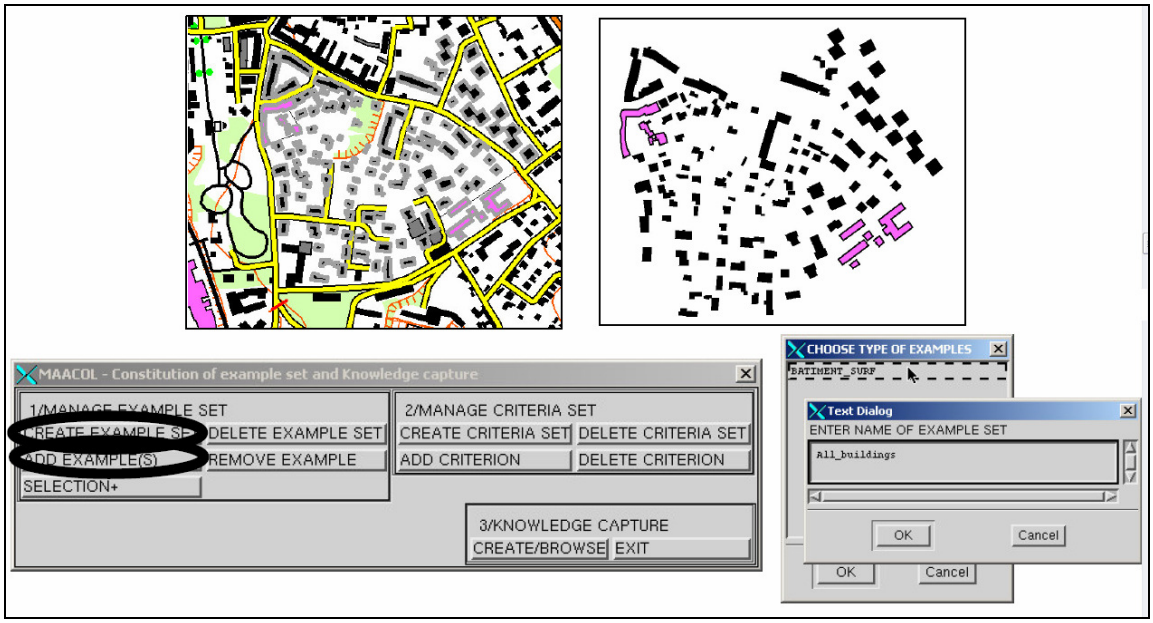


Figure B.6 Création des bases d'exemples avec MAACOL. D'après (Duchêne et al., 2005)

Une autre approche pour remédier au problème de la collecte d'exemples consiste à acquérir directement les exemples par l'expérimentation sans recourir à l'intervention d'experts. Les systèmes basés sur ce type d'approche cherchent à analyser des généralisations passées pour en déduire des connaissances.

(Burghardt & Neun ; 2006) utilisent la technique du filtrage collaboratif pour guider le choix des actions à appliquer sur les objets géographiques. Le moteur du processus de généralisation est présenté figure B.7.

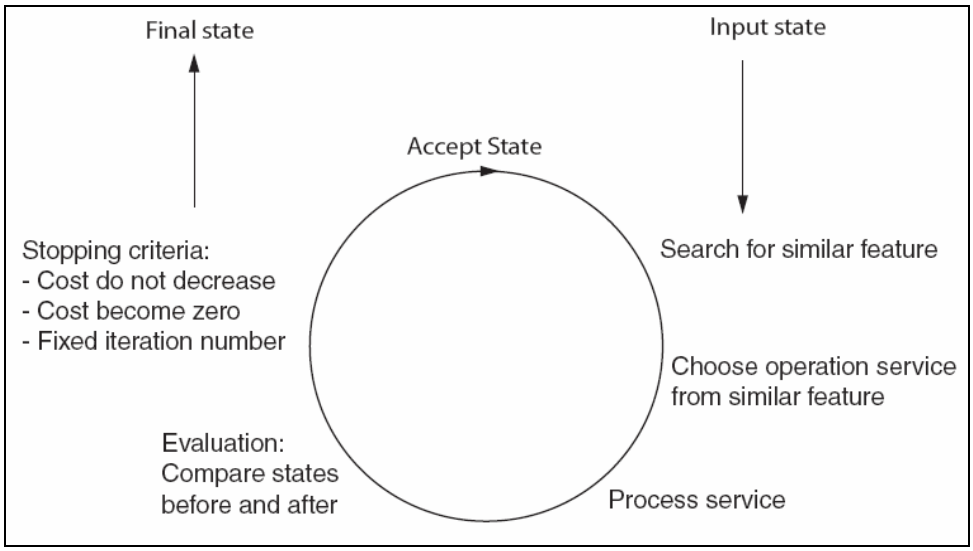


Figure B.7 Moteur du processus de généralisation proposé par (Burghardt & Neun, 2006)

Afin de généraliser un objet géographique, le système applique sur ce dernier une série d'actions de généralisation. Le choix de l'action à appliquer à chaque étape du processus est guidé par les résultats obtenus par les différentes actions pour des objets similaires. Le terme « objet similaire » désigne ici un objet qui se trouve dans un état semblable du point de vue des satisfactions des contraintes cartographiques de l'objet considéré. L'évaluation des résultats obtenus après application d'une action est calculée par comparaison de la satisfaction des contraintes, avant et après application de l'action. Pour un état donné, plus une action améliore la satisfaction des contraintes, plus celle-ci a de chances d'être proposée pour un état similaire.
Initialement, lorsqu'aucune généralisation n'a été effectuée, toutes les actions sont proposées pour chacun des états. Au fur et à mesure que des généralisations sont effectuées, le système tend à ne proposer que les actions permettant l'amélioration a priori la plus importante de l'état courant. On trouve ici une différence par rapport au modèle AGENT qui peut permettre, en fonction du critère de validité des états choisi, des détériorations de l'état courant en vue de ne pas rester bloqué à un optimum local. Une autre différence concerne le mode de représentation des connaissances : dans le modèle AGENT, la plupart des connaissances procédurales sont exprimées sous forme de règles, ici les connaissances procédurales sont exprimées sous forme d'une base de cas.

Une dernière approche visant à acquérir des connaissances procédurales par l'expérience a été proposée par (Ruas et al., 2006). Cette approche, dédiée au modèle de généralisation AGENT, se base sur l'analyse des traces d'exécution pour apprendre de nouvelles règles d'application des actions qui sont ensuite directement ajoutées aux connaissances initiales du système. Nos travaux s'inscrivent dans la continuité de cette approche. Toutefois, nous ne cherchons pas dans notre cas à ajouter directement de nouvelles règles dans le système, mais à réviser les connaissances initiales en conservant leur formalisme de représentation (cf. B.III.2).

B.II.3 Bilan

Nous avons présenté, dans cette partie B.II, de nombreux travaux portant sur l'acquisition et la révision des connaissances en intelligence artificielle mais également dans le domaine de la généralisation de données géographiques.

Savoir réviser des connaissances implique en premier lieu de savoir en apprendre de nouvelles. Une méthode pour réaliser cet apprentissage est d'utiliser les techniques de l'apprentissage artificiel qui permettent de construire automatiquement des connaissances à partir de l'expérience. Nous avons présenté, dans cette partie, les notions de base de l'apprentissage artificiel. Nous nous sommes particulièrement intéressés à l'apprentissage supervisé que nous utiliserons dans le cadre de cette thèse.

Une fois ces notions de base posées, nous avons présenté un court état de l'art des travaux portant sur la révision des connaissances en intelligence artificielle. Ces travaux, de natures extrêmement diverses et prenant pour cadre différents formalismes de représentation des connaissances, ne répondent pas directement à notre problématique de thèse mais donnent des clefs pour y faire face. En effet, si certains de ces travaux proposent des approches pour réviser une base de connaissances à partir de nouvelles informations (base d'exemples, nouveaux faits, etc.), ils ne s'intéressent pas ou peu à la construction de ces nouvelles informations par l'expérience. De plus, le cadre particulier des connaissances que nous cherchons à réviser (connaissances incluses dans des systèmes fonctionnant par exploration informée d'arbres d'états) amène des problématiques spécifiques qui ne sont pas traitées dans ces travaux.

Les techniques de l'apprentissage artificiel ont parfois été utilisées dans le domaine de la généralisation de données géographiques. Nous avons présenté dans une dernière partie, un état de l'art des travaux qui proposent d'utiliser ces techniques dans le cadre de l'automatisation du processus de généralisation. Beaucoup de modèles de généralisation sont basés sur l'utilisation de connaissances procédurales et sont, de fait, particulièrement sensibles aux problèmes liés à la gestion des connaissances. Plusieurs travaux ont mis en œuvre des processus d'acquisition et de révision des connaissances procédurales. Parmi ces derniers, très peu proposent d'acquérir ou de réviser les connaissances de manière totalement automatique. En effet, la plupart reposent sur la participation d'experts dans le processus d'acquisition ou de révision des connaissances (Reichenbacher, 1995 ; Weibel et al., 1995 ; Mustière, 2001). Certains travaux se démarquent toutefois : Ainsi (Burghardt & Neun, 2006) proposent d'utiliser une base de cas construite automatiquement pour guider le choix des actions à appliquer sur les objets géographiques. On peut noter ici une différence par rapport à notre travail de thèse dans lequel nous cherchons à conserver le formalisme de représentation des connaissances du système et à ne pas imposer une représentation particulière (ici, une base de cas). (Ruas et al., 2006) proposent, pour leur part, une approche générale qui répond en partie à nos attentes et que nous reprendrons dans le cadre de cette thèse. Toutefois, nous cherchons dans notre cas à formuler une approche plus générique et qui permet non seulement d'ajouter de nouvelles règles dans le système, mais également de modifier et de supprimer celles déjà existantes.

B.III Révision par l'expérience des connaissances procédurales d'un système fonctionnant par exploration informée d'arbres d'états : formalisation

B.III.1 Système fonctionnant par exploration informée d'arbres d'états

Nous avons introduit au chapitre A, l'objectif applicatif de ce travail de thèse : celui de proposer un modèle de révision automatiquement des connaissances pour les systèmes de généralisation fonctionnant par exploration informée d'arbres d'états. Nous avons ensuite généralisé cet objectif applicatif à l'ensemble des systèmes fonctionnant par exploration informée d'arbres d'états. Ces systèmes ont pour but de résoudre des problèmes d'optimisation.

Nous nous intéressons, dans le cadre de cette thèse, à un type particulier de problèmes d'optimisation que nous présenterons en partie B.III.1.1. Une fois ces problèmes introduits, nous proposerons, en partie B.III.1.2, une formalisation des systèmes destinés à résoudre ces problèmes et dont nous chercherons à réviser les connaissances.

B.III.1.1 Problème d'optimisation

Nous nous intéressons dans le cadre de cette thèse à un type particulier de problème d'optimisation.

Nous rappelons, en préambule, la notion de *problème* et *d'instance* de problème. Un exemple de *problème* peut être : soit *X*, un ensemble de points, trouver un plus court chemin passant par tous les points de *X*. Une *instance* de ce problème peut être : soient les points $\{x_1, x_2, x_3\}$, trouver un plus court chemin passant par ces trois points.

Nous proposons pour le type de problème d'optimisation qui nous intéresse, la définition suivante :
Soit *P*, un problème d'optimisation caractérisé par :

- Ent_P : classe d'entités. Chaque entité *ent* est caractérisée par un état *e*.
- M_P : ensemble des états pouvant être pris par l'ensemble des entités de classe Ent_P. M_P peut être infini.
- $\{Action\}_P$: ensemble fini d'actions. L'application d'une action sur une entité permet le passage de celle-ci d'un état *e* à un état *e'*. L'application des actions est supposée non prévisible, c'est-à-dire que les conséquences de l'application d'une action ne sont pas connues : si l'état *e'* résulte de l'application d'une action sur une entité se trouvant dans un état *e*, la connaissance de *e* ne nous permet pas de prévoir l'état *e'*. Nous noterons dans la suite de cette thèse, *Act(e)*, pour désigner l'état résultant de l'application de l'action *Act* sur une entité se trouvant dans un état *e*.
- $Q_P(e)$: évaluation d'un état $e \in M_P$

Une instance *p* de *P* consiste à déterminer, pour une entité *ent* de classe Ent_P se trouvant dans un état initial donné, une séquence d'actions permettant d'obtenir un état du monde m_p qui maximise la fonction Q_P. Le monde $m_p(ent)$ est un sous ensemble de M_P. Il représente

l'ensemble des états qu'il est possible d'obtenir par application d'une quelconque séquence d'actions à partir de l'état initial de l'entité *ent* considérée. Une action peut être employée plusieurs fois par séquence d'actions, le nombre d'états contenus dans m_p peut donc être infini. La figure B.8 illustre ces définitions.

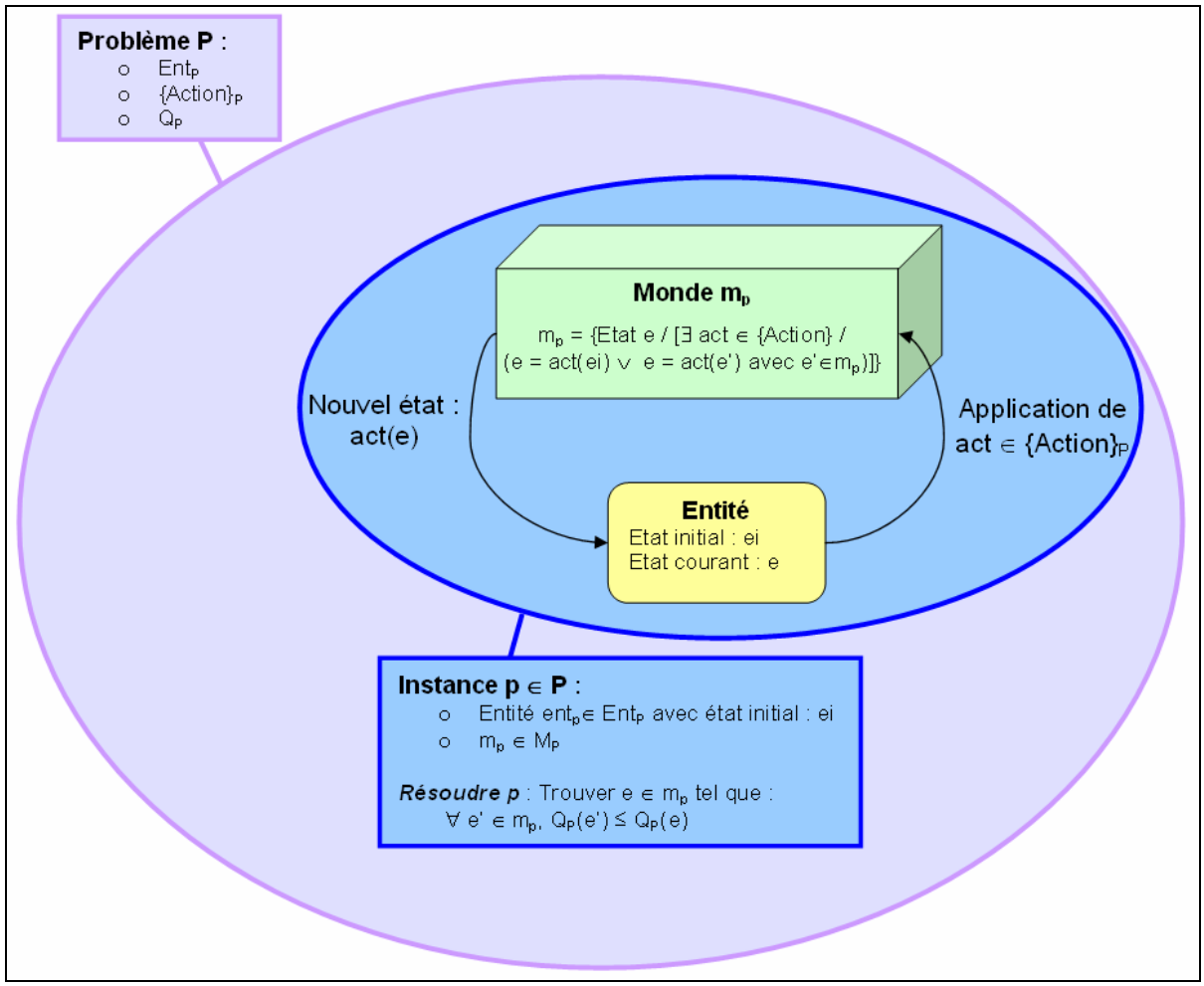


Figure B.8 Problèmes d'optimisation traités

Pour donner un exemple, la généralisation des objets géographiques de la classe *bâtiment* par le modèle AGENT peut être vue comme un problème *Pbat* où :

- Ent_P est la classe *bâtiment*.
- M_P est l'ensemble des états pouvant être pris par les objets de la classe *bâtiment*.
- $\{Action\}_{Pbat}$ = {Grossissement, Grossissement en rectangle, Simplification, Equarrissage}
- $Q_{Pbat}(e)$ est la fonction de calcul de la satisfaction d'un état d'un agent *bâtiment*.

La généralisation d'un bâtiment précis appartenant à la classe *bâtiment*, est une instance particulière de *Pbat*, où l'état initial désigne l'état initial du bâtiment.

Ce type de problème d'optimisation peut se rapprocher des problèmes traités en apprentissage par renforcement (Bellman, 1957 ; Sutton, 1988 ; Watkins. & Dayan, 1992). Un problème d'apprentissage par renforcement pour un agent se définit ainsi :

- Un ensemble d'états *S* pouvant être pris par l'agent
- Un ensemble *A* d'actions possibles pour l'agent
- Une fonction de récompense $R : \{S,A\} \rightarrow \rho$

L'agent choisit, à chaque pas de temps, une action de *A* en fonction de son état courant. A chaque application d'une action, l'agent reçoit une récompense.
L'évolution de l'environnement est assurée par un processus de décision markovien (MDP). Ce dernier est défini par :

- Une fonction de transition qui donne la probabilité de passer d'un état *s'* à un état *s* par application d'une action *act*.
- Une fonction de récompense qui donne une récompense lorsque l'agent passe d'un état *s* à un état *s'* par application d'une action *act*.

Le but de l'apprentissage par renforcement est de trouver une politique qui maximise la récompense à long terme. Une politique a pour rôle de spécifier, pour chaque état *s*, la probabilité d'appliquer chacune des actions. Le choix d'une action est uniquement dépendant de l'état courant.

Un point clef de l'apprentissage par renforcement est qu'il permet d'améliorer les connaissances en ligne (à chaque instance de problème résolue, la politique est améliorée). Cette caractéristique peut, en effet, s'avérer indispensable pour certaines applications.

Dans notre problématique de révision des connaissances, où nous cherchons à conserver le formalisme de représentation des différentes connaissances, l'apprentissage par renforcement pourrait être utilisé, dans certains cas, pour les connaissances de construction et d'ordonnancement de la liste d'actions. Cette utilisation nécessite toutefois de pouvoir traduire les connaissances du système sous la forme d'une politique et vice versa. Or, cette traduction peut s'avérer complexe, et parfois même impossible dans le cas de connaissances qui ne dépendent pas uniquement de l'état courant de l'agent.

Dans le cadre de ce travail de thèse, où l'aspect « amélioration en ligne », n'est pas une contrainte imposée (cf. B.V.1) et où nous ne posons pas de contraintes sur le formalisme de représentation des connaissances, l'utilisation de l'apprentissage par renforcement ne paraît pas pertinent. Nous ne l'utiliserons donc pas dans ce travail de thèse.

B.III.1.2 Système de résolution de problèmes fonctionnant par exploration informée d'arbres d'états

Nous avons défini dans la partie précédente le type de problème d'optimisation auquel sont dédiés les systèmes qui vont nous intéresser dans le cadre de cette thèse. Nous allons maintenant définir les systèmes en eux-mêmes.

Un *système fonctionnant par exploration informée d'arbres d'états* peut permettre de résoudre des problèmes d'optimisation tels que ceux décrits en partie B.III.1.1. Il se base, pour les résoudre, sur deux principes :

- *Exploration d'arbre d'états* : un arbre d'états est construit par instance du problème. Chaque nœud de l'arbre représente un état du monde (de l'entité). Le passage d'un nœud à un autre correspond à l'application d'une action.

- *Exploration informée* : la stratégie d'exploration des arbres d'états est basée sur l'utilisation d'un ensemble de connaissances procédurales (heuristiques) $\{K_{procédurale}\}$ propres à un problème considéré. Chaque type de connaissances composant $\{K_{procédurale}\}$ peut être représenté dans un formalisme différent.
  Chaque connaissance procédurale est caractérisée par un domaine de définition, qui représente l'ensemble des valeurs que la connaissance peut prendre, et par une valeur prise sur ce domaine de définition. Par exemple, un domaine de définition possible pour une connaissance d'application d'une action *Act* peut être : {toujours proposer *Act*, ne jamais proposer *Act*}. La connaissance d'application de *Act* peut prendre la première valeur de son domaine de définition, et ainsi toujours proposer l'action *Act* quel que soit l'état de l'entité, ou prendre la seconde valeur de son domaine de définition, et alors ne jamais proposer *Act*. La connaissance d'application de l'action *Act* ne peut prendre que l'une de ces deux valeurs.
  Le domaine de définition d'une connaissance est directement lié à son formalisme de représentation et peut être infini. Nous appelons *jeu de connaissances* l'ensemble des connaissances procédurales d'un système : une valeur de son domaine de définition est affectée à chaque connaissance.

Par exemple, dans le cas du modèle AGENT :

- *Exploration d'arbre d'états* : le modèle AGENT, fonctionne par exploration en profondeur de type « meilleur d'abord » d'un arbre d'états. Un arbre d'états est construit par objet géographique à généraliser. Un état du monde correspond à un état de l'objet géographique.
- *Exploration informée* : les connaissances procédurales du modèle AGENT sont les suivantes (partie A.3.4) : {connaissances sur les priorités des contraintes, connaissances sur les applications des actions}.

L'enjeu de tels systèmes est de pouvoir résoudre les problèmes d'optimisation décrits en partie B.III.1.1 en un temps acceptable. Le système doit donc trouver un état satisfaisant en parcourant le moins d'états possible.

Nous proposons, en figure B.9, un cycle d'actions générique pour le système. Le principe de ce cycle d'actions est le suivant : le système caractérise et évalue, dans une première étape, l'état de l'entité traitée en utilisant la fonction Q_P. Il teste ensuite le critère de fin de cycle. Si ce critère est vérifié, le cycle d'actions s'arrête, dans le cas contraire, le système passe à l'étape suivante. Le système teste ensuite la validité de l'état de l'entité : si l'état est valide, le système utilise une heuristique pour créer et ordonner une liste d'actions pouvant être appliquées pour cet état de l'entité ; dans le cas contraire, le système fait revenir l'entité à son état précédent. Le système vérifie ensuite s'il reste des actions dans la liste d'actions. Si ce n'est pas le cas, le système faire revenir l'entité à son état précédent. Si l'entité revient à son état initial et qu'il n'y a plus d'action à appliquer pour cet état, le cycle d'actions s'arrête. Dans le cas où la liste d'actions n'est pas vide, le système choisit la meilleure action à appliquer, l'applique sur l'entité et retire l'action de la liste d'actions puis revient à la première étape du cycle d'actions avec un nouvel état de l'entité.

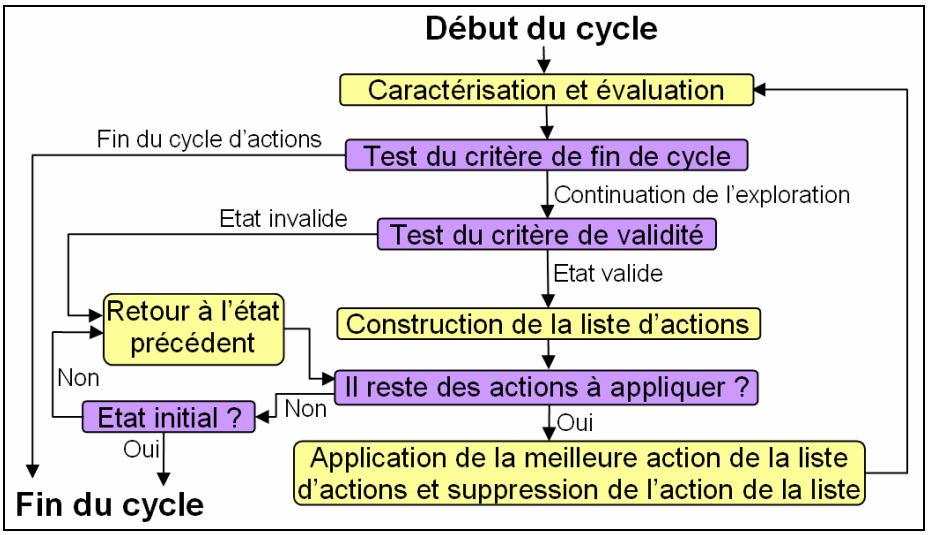


Figure B.9 Cycle d'actions générique

Trois types de connaissances interviennent dans ce cycle d'actions : les connaissances de construction et d'ordonnancement de la liste d'actions, le critère de validité et le critère de fin de cycle.

Le moteur du modèle AGENT est totalement compatible avec ce cycle d'actions générique. Dans son cas, les connaissances de construction et d'ordonnancement de la liste d'actions sont définies au travers des connaissances de priorité des contraintes et d'application des actions.

B.III.2 Révision par l'expérience des connaissances procédurales

Nous avons proposé en partie B.III.1, une définition des problèmes d'optimisation et des systèmes fonctionnant par exploration informée d'arbres d'états qui nous intéressent dans le cadre de cette thèse.

La stratégie d'exploration employée par ces systèmes a une influence directe sur leurs performances. Nous proposons donc en partie B.III.2.1 de nous intéresser à l'évaluation des performances d'un tel système. Nous présentons ensuite, en partie B.III.2.2, une reformulation du problème de la révision des connaissances en tant que problème de recherche, à partir d'un jeu de connaissances initial et d'un jeu de connaissances qui maximise les performances du système.

B.III.2.1 Evaluation des performances d'un système fonctionnant par exploration d'arbres d'états

La question de l'évaluation des performances d'un système est souvent liée à la nature même du système. Si quelques travaux tels que ceux de (Suganthan et al., 2005 ; Taillard, 2005) s'intéressent à la comparaison de méthodes de résolution de problèmes, peu donnent des méthodes d'évaluation pouvant être directement utilisées dans le cadre de notre problématique.

Nous nous intéressons, pour nos systèmes fonctionnant par exploration informée d'arbres d'états, à 2 critères :

- *L'efficacité* : correspond à la capacité d'un système à amener l'entité dans un bon état (en termes de valeur de Q_P).
- *L'efficience* : correspond à la capacité d'un système à amener rapidement une entité dans son meilleur état. Dans le cas où toutes les actions ont des coûts similaires du point de vue du temps de calcul nécessaire à leur application, l'efficience est directement dépendante du nombre d'états parcourus.

Améliorer l'efficacité du système implique souvent d'explorer plus d'états, ce qui a généralement pour conséquence une diminution de l'efficience du système. Il est donc nécessaire de trouver un compromis entre l'efficacité et l'efficience du système.

Le critère choisi pour évaluer la stratégie d'exploration doit tenir compte des besoins liés à l'utilisation du système. En effet, pour certaines applications, l'efficacité devra être privilégiée, alors que pour d'autres, c'est l'efficience qui devra l'être.

Nous notons $m_p(S)$ l'ensemble des états parcourus par un système S pour résoudre une instance p d'un problème d'optimisation P. $m_p(S)$ est un sous-ensemble de m_p. Nous notons de la même manière $meilleur\_etat_p(S)$ le meilleur état trouvé par un système S dans le cadre de la résolution d'une instance p d'un problème d'optimisation P.

Nous définissons *perf(S, p)*, une fonction évaluant les performances d'un système S (et donc de sa stratégie d'exploration), pour une instance p d'un problème d'optimisation P. Cette fonction est liée à l'efficience et à l'efficacité du système pour cette instance. Un exemple de fonction *perf* est :

$$perf(S, p) = \frac{Q_P(meilleur\_etat_p(S))}{card(m_p(S))}$$

Nous avons imposé, dans nos hypothèses, qu'une stratégie d'exploration soit définie par problème. On peut donc généraliser notre fonction d'évaluation des performances à un problème. Nous posons ainsi la fonction *Perf(S, P)* qui évalue les performances d'un système S pour un problème d'optimisation P. Un exemple de fonction *Perf (S, P)* pour un problème composé d'un nombre fini d'instances peut être la moyenne arithmétique des performances du système pour chaque instance de P:

$$Perf(S, P) = \frac{1}{card(P)} \sum_{p \in P} perf(S, p)$$

Le choix de la fonction *Perf(S, P)* dépendra, à l'image de la fonction *perf(S, p)*, des besoins sous-tendant l'utilisation du système.

Dans de nombreux cas, les problèmes à traiter sont composés d'un nombre très important d'instances, voire infini. Il peut donc être très difficile, voire impossible, de calculer cette fonction d'évaluation.

Nous nous proposons alors de définir P_n, un échantillon de n instances de P. On peut alors calculer $Perf(S, P_n)$ qui évalue les performances d'un système S pour l'échantillon P_n constitué de n instances de P. Nous allons nous servir de $Perf(S, P_n)$ pour fournir une estimation empirique de la valeur de la fonction $Perf(S, P)$.

Définir un système optimal pour $Perf(S, P_n)$, ne permet pas de garantir que ce même système est aussi optimal pour $Perf(S, P)$.

Nous nous retrouvons face à des difficultés similaires à celles rencontrées dans le cadre de l'apprentissage artificiel supervisé, à savoir le risque de surapprentissage. Un point important de l'estimation, par la fonction $Perf(S,P_n)$, de la performance d'un système, pour un problème d'optimisation P, est donc de bien choisir l'échantillon d'instances utilisé pour l'évaluation, afin que celui-ci soit réellement représentatif de l'ensemble des instances du problème.

B.III.2.2 Objectif de la révision

Nous avons présenté dans la partie précédente une fonction d'évaluation permettant de fournir une évaluation des performances d'un système, et donc de sa stratégie d'exploration, pour un problème donné.

Dans le cadre de systèmes basés sur l'exploration informée tels que ceux qui nous intéressent, la stratégie d'exploration est définie à partir de connaissances procédurales.
Nous proposons donc d'utiliser la fonction $Perf(S, P_n)$ pour estimer la pertinence d'un jeu de connaissances procédurales vis-à-vis d'un système S et d'un problème d'optimisation P avec P_n un échantillon représentatif de n instances de P.

Nous notons S_K un système S fonctionnant par exploration informée d'arbres d'états et dont la stratégie d'exploration est basée sur un jeu de connaissances K.

Le problème de la révision des connaissances revient donc à trouver, à l'aide du jeu de connaissances initial, le jeu K, parmi l'ensemble des jeux de connaissances possibles, qui permet d'optimiser la fonction $Perf(S_K, P_n)$.

On se retrouve face au même problème que celui évoqué dans la partie précédente : trouver un jeu de connaissances optimal pour un échantillon d'instances du problème n'assure pas que celui-ci le soit réellement pour l'ensemble des instances. Un risque est de se retrouver confronté au problème classique en apprentissage artificiel du surapprentissage. Le choix de l'échantillon d'instances utilisé pour réviser les connaissances est donc primordial.

B.IV Révision par l'expérience des connaissances procédurales d'un système fonctionnant par exploration informée d'arbres d'états : problématiques

B.IV.1 Problèmes liés au diagnostic des connaissances

Un aspect important d'un processus de révision des connaissances concerne le diagnostic des connaissances. En effet, une première étape avant de procéder à une révision effective des connaissances est de déterminer si ces dernières méritent ou non d'être révisées.

Nous avons présenté, dans la partie précédente, une fonction permettant d'obtenir une estimation de la qualité d'un jeu de connaissances. Cette fonction peut se révéler insuffisante pour prendre une décision pertinente concernant la nécessité ou non de réviser les connaissances. En effet, la fonction d'évaluation ne fait que donner une évaluation globale des performances du système pour un jeu donné de connaissances, elle ne fournit pas d'informations précises sur la part de connaissances méritant d'être révisées.

Une stratégie pour détecter ces dernières est de passer par une analyse locale, au niveau de chacune des connaissances du système. Une difficulté soulevée par ce type d'analyse provient de l'interdépendance des connaissances entre elles. En effet, les succès et les échecs obtenus par une connaissance peut dépendre des autres connaissances : typiquement, pour un état donné, l'application d'une action peut entraîner l'exploration de sous-arbres très différents en fonction de l'élagage choisi. La figure B.10 illustre ce problème d'interdépendance : dans le cas de gauche, où le critère de validité admet des détériorations de la qualité des états, il est recommandé d'utiliser *l'Action 1* lorsque l'on se trouve dans *l'état 1*. L'application de celle-ci a, en effet, permis après application de *l'Action 2* de trouver *l'état 3* qui est le meilleur pour cet arbre. En revanche, si le critère de validité n'admet pas les détériorations comme dans le cas de droite, il est alors recommandé d'utiliser *l'Action 2* à *l'état 1* de façon à tomber directement sur *l'état 4* qui est le meilleur pour cet arbre.

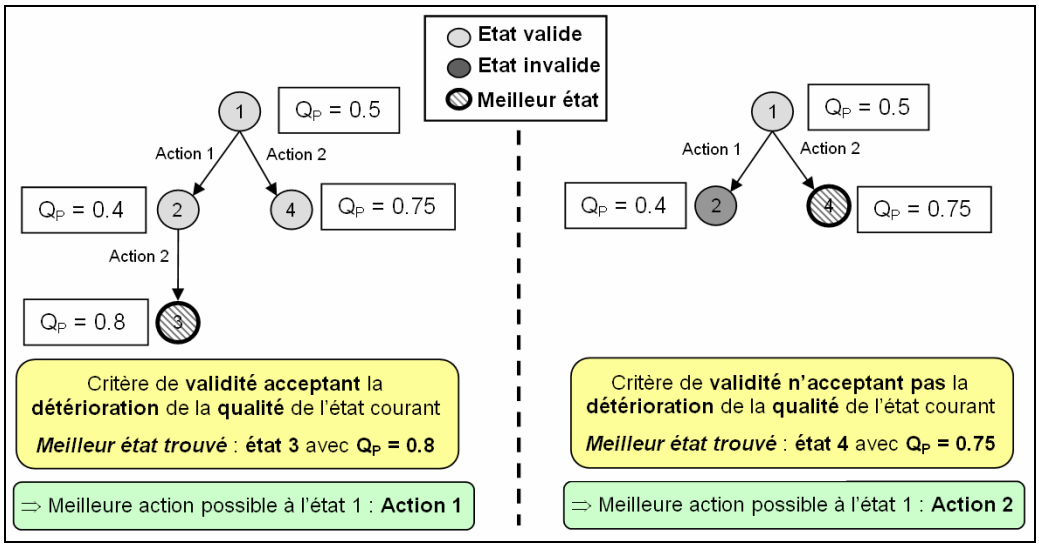


Figure B.10 Exemple d'interdépendance entre les connaissances

Les outils d'analyse des connaissances constituent un premier pré-requis pour la prise de décision concernant la nécessité ou non d'une révision des connaissances. Une fois ces outils définis, il reste à proposer un processus complet de prise de décision permettant de définir, à partir des informations tirées des analyses locales et globales, si le jeu de connaissances mérite d'être révisé ou non.

Une difficulté, déjà évoquée en partie B.III.2.1, de cette prise de décision concerne le problème de la représentativité des analyses qui peuvent être faites. En effet, si un jeu de connaissances relatif à un problème d'optimisation *P*, se révèle mauvais pour un échantillon d'instances de *P*, peut-on assurer pour autant que c'est aussi le cas pour l'ensemble des instances de *P* ? L'échantillon d'instances de *P* choisi pour évaluer un jeu de connaissances peut très bien n'être composé que de cas particuliers ne reflétant pas le cas général. Une question importante à se poser est donc de savoir à quel point un diagnostic réalisé à partir d'un échantillon d'instances est représentatif du cas général.

Diagnostiquer et déterminer les besoins en révision, nécessite de répondre à l'ensemble de ces questions. Nous reviendrons longuement sur cette partie diagnostic dans le chapitre E.

B.IV.2 Problèmes liés à l'acquisition de connaissances

L'enjeu de ce travail de thèse est de réviser les connaissances par l'expérience sans intervention d'experts. Le processus de révision doit donc, par analyse d'instances du problème déjà résolues, tirer de nouvelles informations sur leur résolution afin d'améliorer le jeu de connaissances utilisé par le système.

Nous nous intéressons dans cette partie B.IV.2 aux questions liées à l'acquisition de nouvelles connaissances à partir de l'expérience.

Une première question à se poser concerne le choix de la représentation à adopter pour formaliser l'expérience passée ainsi que le choix des informations à en tirer.

Répondre à cette question nécessite dans un premier temps de bien définir le type de connaissances que l'on souhaite apprendre. Ces connaissances dépendent du système considéré. Elles ont néanmoins pour point commun de toutes avoir traits à la stratégie d'exploration des arbres d'états.

Un point capital dans la formalisation de l'expérience concerne le choix du langage de description des informations tirées des instances du problème déjà traitées. Ce choix demande en particulier de définir comment les états des arbres d'états seront caractérisés. Il faut que le langage de description soit assez riche pour pouvoir décrire correctement les états et ainsi éviter les bruits de description. En effet, un langage de description trop pauvre ne permet pas d'appréhender tous les concepts nécessaires à l'apprentissage. Un risque est alors de se trouver avec des informations incohérentes : typiquement, deux états différents et nécessitant l'application d'actions différentes, décrits exactement de la même manière dans le langage de description. Un langage trop riche, pour sa part, peut entraîner une explosion de la taille de l'espace des hypothèses et avoir aussi des conséquences négatives sur le résultat de l'apprentissage (Neri & Saitta, 1994). En effet, un langage trop riche peut rendre l'information utile très implicite et donc difficilement appréhendable par les algorithmes d'apprentissage. (Caruana & Freitag, 1994) exposent ainsi deux raisons pour lesquelles un excès d'attributs peut détériorer les performances de l'apprentissage : la première provient de

la présence d'attributs bruités, la seconde des interactions pouvant exister entre un attribut et le biais utilisé par l'algorithme d'apprentissage.
Dans le cadre de nos travaux, où l'on souhaite réviser des connaissances et ne pas simplement en acquérir de nouvelles, le langage de description sera souvent imposé par le type de connaissances considéré.

Une seconde question concerne le choix des informations à utiliser pour acquérir de nouvelles connaissances. Nous nous plaçons, dans le cadre de cette thèse, dans le cas où ces informations sont extraites des instances du problème d'optimisation déjà traitées. En raison du caractère non prévisible des actions, il n'est pas possible de prédire à l'avance sur quoi porteront les informations qui pourront être tirées de la résolution d'une instance. On ne peut pas choisir directement les informations souhaitées mais simplement les instances à traiter. Nous avons déjà évoqué les difficultés liées aux choix des instances du problème d'optimisation : ces dernières doivent être représentatives de l'ensemble des instances du problème d'optimisation si l'on souhaite que les informations pouvant être tirées de leur résolution soient réellement pertinentes. Nous reviendrons sur cette question en partie C.II.

Une dernière question concerne la validation des jeux de connaissances appris. Il est, en effet, important d'assurer que celui-ci soit réellement meilleur que le jeu de connaissances initial. Nous avons présenté dans la partie B.III.2.1, une fonction permettant de fournir une estimation de la qualité d'un jeu de connaissances. Cette fonction doit être définie en fonction du domaine d'application et des besoins sous-tendant l'utilisation du système. Quelle que soit la fonction définie, sa pertinence sera directement dépendante de la représentativité de l'échantillon d'instances choisi pour l'évaluation. On retombe donc sur le problème du choix d'instances représentatives qui sera traité en partie C.II.

B.IV.3 Problèmes liés à la révision des connaissances

Nous ne cherchons pas, dans ce travaille de thèse, à simplement acquérir de nouvelles connaissances, mais à réviser celles déjà existantes. Cet aspect « révision » amène ses propres questions.

En effet, procéder à une révision implique de tenir compte des valeurs initiales des connaissances, mais comment tenir compte de celles-ci dans le processus de révision ?

Faut-il donner une grande importance au jeu de connaissances initial quitte à passer à coté d'un très bon jeu de connaissances appris si ce dernier n'est pas compatible avec le jeu initial ? Faut-il, au contraire, ne pas tenir compte du jeu de connaissances initial, et par conséquent ne pas profiter des informations potentiellement intéressantes qu'il peut inclure ? La question est donc de savoir où situer le curseur entre ces deux extrêmes.

L'intérêt de procéder à une révision des connaissances et non à une acquisition est de profiter des informations incluses dans le jeu de connaissances initial. Parfois, l'analyse d'instances déjà traitées ne suffit pas à percevoir tous les aspects de la résolution de l'ensemble des instances du problème. Typiquement, lorsque la fonction d'évaluation des états n'est pas parfaite et ne permet pas de retranscrire exactement la qualité réelle d'un état, le jeu de connaissances initial peut, lui, tenir compte de ces défauts et ainsi apporter des éléments extérieurs importants pour la qualité du jeu de connaissances final. De même, le jeu de connaissances initial peut tenir compte de l'instabilité de certaines actions qui peut provoquer

des bogues du système, des fuites de mémoire ou des problèmes de caractérisation des états. Il est donc intéressant de pouvoir tenir compte, au moins dans une certaine mesure, du jeu de connaissances initial.

B.V Approche et modèle généraux de révision des connaissances proposés

B.V.1 Approche générale proposée pour la révision des connaissances procédurales

B.V.1.1 Introduction

Nous avons posé en partie B.IV, les problèmes liés à la révision, par l'expérience, des connaissances dans un système fonctionnant par exploration informée d'arbres d'états. Nous allons maintenant exposer notre approche générale pour la révision des connaissances.

Notre approche générale est basée sur l'analyse d'instances du problème déjà traitées, autrement dit sur l'analyse de traces. Nous avons déjà évoqué en partie A.V.2 l'intérêt de pouvoir réviser les connaissances du système totalement automatiquement. En effet, recourir à des experts pour qu'ils étiquettent des données d'apprentissage peut se révéler très fastidieux, comme l'ont montré (Mustière, 2001 ; Ruas & Holzapflel, 2003) dans le cadre de l'automatisation de la généralisation. Il devient donc très intéressant de disposer d'un système capable, par introspection, de corriger de lui-même ses propres connaissances sans qu'un expert ait à intervenir. Nous proposons donc de tracer le système pendant qu'il traite des instances du problème d'optimisation considéré et de nous servir de ces traces pour réviser les connaissances du système.

Notre objectif applicatif ne nous impose pas de réviser les connaissances en ligne. Nous pourrons donc avoir recours à une révision hors ligne des connaissances. Cet aspect hors ligne permet de dissocier la phase de révision du fonctionnement normal du système (figure B.11), et ainsi de faire face aux difficultés évoquées en partie B.IV.1 concernant la représentativité des instances utilisées pour la révision et les dépendances entre connaissances. Nous revenons sur ce point en partie B.V.1.2.

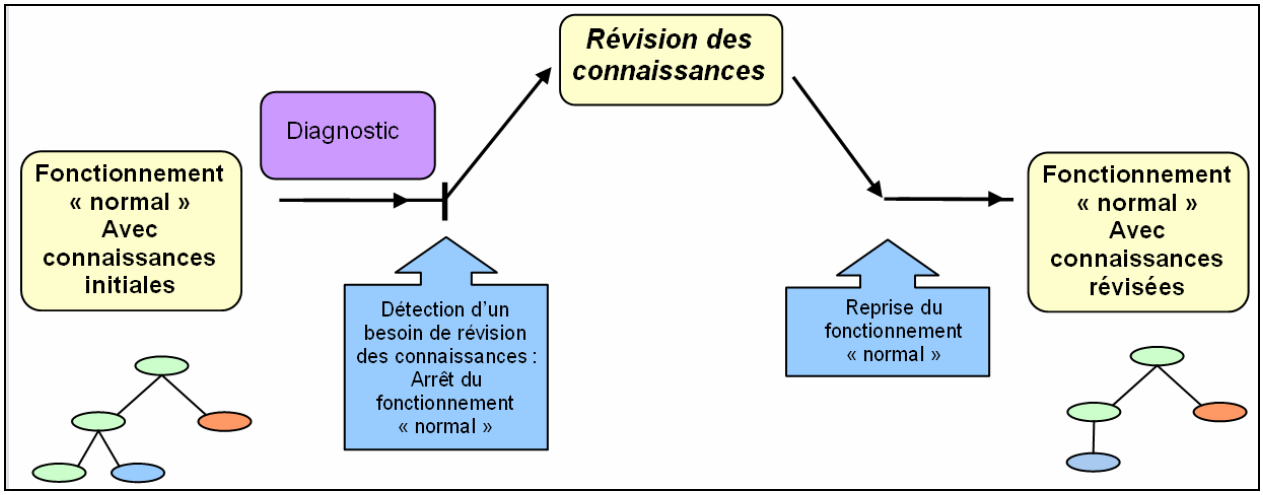


Figure B.11 Approche générale de révision des connaissances proposée

Nous proposons d'intégrer au système un module de diagnostic qui permet, lors d'une utilisation normale du système, de définir si le jeu de connaissances nécessite ou non d'être révisé. Dans le cas où il nécessite d'être révisé, le fonctionnement normal du système est interrompu et le système déclenche le processus de révision des connaissances. Une fois cette phase de révision terminée, le système reprend son fonctionnement normal avec le jeu de connaissances révisé. Nous revenons rapidement sur ce module de diagnostic en partie B.V.1.3.

B.V.1.2 Processus de révision des connaissances

Le processus de révision est déclenché lorsque le module de diagnostic détecte une « non optimalité » probable des connaissances. Il se déroule hors ligne.

L'intérêt de procéder à une révision hors ligne est double : d'une part, elle permet au processus de révision d'avoir un contrôle complet du choix des instances du problème d'optimisation qui seront utilisées comme base d'expérience, et d'autre part, elle permet de résoudre les instances avec un jeu de connaissances plus adapté.

Pour revenir sur le premier point, nous avons déjà évoqué l'enjeu du choix des instances considérés lors du processus de révision : ce dernier à une importance capitale sur la qualité des connaissances qui peuvent être apprises. Réviser les connaissances en ligne, c'est-à-dire instance par instance et sans contrôle sur le choix des instances proposées, peut avoir des effets néfastes sur les connaissances si les instances traitées font figure de cas particuliers pour le problème d'optimisation considéré.

Concernant le second point, nous avons évoqué en partie B.IV.1 les problèmes liés à l'interdépendance entre connaissances. Les connaissances sont liées entre elles, ce qui entraîne l'introduction d'un biais lorsque l'on souhaite réaliser une étude locale, au niveau de chacune des connaissances. Ce biais est d'autant plus important que les connaissances utilisées pour définir l'élagage des arbres limitent l'exploration de ces derniers. Ainsi un arbre très élagué n'apportera que peu d'informations sur les connaissances nécessaires au traitement de ce type de problème d'optimisation.

Le processus de révision se décompose en deux phases :

- **Phase d'exploration** : cette première phase consiste à tracer le processus pendant qu'il traite un échantillon d'instances du problème d'optimisation considéré. Le module de révision doit donc choisir, dans un premier temps, les instances du problème d'optimisation qui composeront l'échantillon de révision, puis dans un second temps, traiter ces instances avec des connaissances permettant d'en tirer le plus d'informations possible. Nous reviendrons sur cette phase de constitution de l'expérience dans le chapitre C.
- **Phase d'analyse** : cette seconde phase consiste à utiliser l'échantillon d'instances traitées durant la précédente phase afin de réviser les connaissances. La révision se base sur l'analyse des instances traitées afin d'en extraire des informations importantes pour les connaissances. Nous reviendrons sur cette phase dans le chapitre D.

B.V.1.3 Module de diagnostic

Le module de diagnostic a pour rôle d'évaluer en ligne les connaissances et peut proposer de déclencher le processus de révision des connaissances lorsque nécessaire.

Comme nous l'avons évoqué en partie B.IV.1, le diagnostic des connaissances pose de nombreux problèmes qui sont, en premier lieu, dus à l'interdépendance des connaissances entre elles. En effet, contrairement à la phase de révision où les connaissances choisies permettront de tirer des informations réellement pertinentes sur les instances du problème d'optimisation analysées, lors de la phase de diagnostic l'analyse des instances sera, dans la plupart des cas, biaisée par l'élagage utilisé.

Le module de diagnostic ne peut donc pas fournir d'évaluation aussi pertinente des connaissances que celle qui peut être réalisée durant le processus de révision. Il doit néanmoins apporter des indications sur la qualité des connaissances et ainsi participer à la prise de décision concernant le déclenchement du processus de révision des connaissances.

Le chapitre E est consacré au module de diagnostic.

B.V.2 Modélisation agent des connaissances

Nous avons présenté dans la partie précédente notre approche générale. Cette approche nécessite, pour être appliquée, de pouvoir modifier dynamiquement chaque connaissance. Un premier point important de la modélisation est donc de proposer un modèle de connaissances modifiable dynamiquement.

Un second point important concerne l'organisation des connaissances. Nous proposons, à cet égard, à l'image des systèmes experts (Buchanan et al., 1969 ; Shortliffe, 1976), d'externaliser les connaissances du système de résolution de problèmes. L'intérêt de cette externalisation est qu'elle permet de faciliter la maintenance des bases de connaissances ainsi que l'adaptabilité du système de résolution de problèmes à différents problèmes d'optimisation.

En plus d'externaliser les connaissances, nous proposons de confier leur gestion à des *agents connaissance*. Chaque connaissance (connaissance de validité, d'application d'une action, etc.) est prise en charge par un *agent connaissance*. Celui-ci a pour rôle de fournir des réponses aux requêtes du système de résolution de problèmes au sujet du contenu de la connaissance qu'il représente, de diagnostiquer la qualité de la connaissance et de participer au processus de révision.

Si l'on reprend les difficultés soulevées en partie B.IV.1, cette modélisation paraît naturelle. En effet, pour une instance donnée du problème d'optimisation traité, une analyse complète des connaissances du système de résolution de problèmes requiert d'analyser localement chaque connaissance. Or, en raison des dépendances existant entre elles, chaque connaissance ne peut être analysée isolément mais doit l'être conjointement à l'ensemble des connaissances avec lesquelles elle partage des liens (cf. B.IV.1).
En plus de son aspect naturel, la modélisation agent a ici d'autant plus d'intérêt que l'environnement est dynamique : effectivement, les *connaissances* doivent être réadaptées à chaque évolution du système, et en particulier lors de l'ajout ou de la suppression de certains éléments tels que des actions (cf. A.V.2). Ainsi, il peut s'avérer fréquent que de nouvelles connaissances apparaissent (exemple : ajout d'une nouvelle action) ou que d'autres disparaissent (exemple : suppression d'une action).

Un *agent connaissance* a trois principaux rôles :

- **Fournir des réponses sur sa connaissance au système** : il doit apporter des réponses aux questions du système qui portent sur la connaissance qu'il représente. Par exemple, un *agent connaissance* représentant la connaissance de validité des états peut répondre à une question du type : *compte tenu du problème traité, l'état courant est-il ou non valide ?*
- **Contrôler la qualité de sa connaissance** : un *agent connaissance* intervient dans le processus de diagnostic. Il peut en effet, par introspection, analyser les succès et les échecs que sa connaissance a rencontrés pour une instance du problème et communiquer les résultats de son analyse à l'*agent diagnostic*.
- **Réviser sa connaissance** : un *agent connaissance* intervient activement dans le processus de révision des connaissances. Dans le cadre de la révision d'un type particulier de connaissances, un *agent connaissance* est en contact direct avec le module de révision et peut être amené à communiquer avec d'autres *agents connaissance*.

Chaque *agent connaissance* possède un *coefficient de qualité des connaissances*. Ce coefficient, qui est compris entre [0,1], caractérise la qualité de la valeur définie pour la connaissance que *l'agent connaissance* représente. Plus il est élevé, moins la valeur de la connaissance sera susceptible d'être modifiée lors du déclenchement d'un processus de révision des connaissances. Une valeur de 0 signifie que l'on se place dans un cadre proche d'une acquisition de connaissances (sans tenir compte de la valeur initiale de la connaissance) alors qu'une valeur de 1 signifie que l'on ne souhaite pas modifier la valeur de la connaissance au cours du processus de révision.
L'intérêt de disposer d'un tel coefficient est de pouvoir jouer sur la prise en compte des valeurs initiales des connaissances lors du processus de révision. Ainsi, si l'on dispose d'une valeur initiale particulièrement fiable pour une connaissance et qui apporte une vraie plus-value par rapport aux informations qui peuvent être tirées d'instances de problème traitées, on peut affecter à l'*agent connaissance* représentant cette connaissance, une valeur de coefficient de qualité proche de 1. Si, par contre, la valeur initiale d'une connaissance n'a a priori pas d'intérêt pour le processus de révision, il est possible de donner à ce coefficient une valeur de 0, et ainsi ne tenir que très peu compte de cette valeur initiale.

Dans le cadre du diagnostic des connaissances, les *agents connaissance* permettent d'avoir une vision locale, au niveau de chacune des connaissances, de la qualité du jeu de connaissances. De façon à disposer également d'une vision globale des résultats, nous introduisons un *agent diagnostic* chargé d'analyser, d'un point de vue global et non plus local, les instances de problème traitées. Un *agent diagnostic* est défini par problème d'optimisation. Par exemple, dans le cadre du modèle AGENT, un agent diagnostic est défini pour la généralisation des agents *bâtiment*, un autre pour la généralisation des agents *route*, etc.. Il joue le rôle du module de diagnostic et participe, avec les *agents connaissance*, à la prise de décision concernant le déclenchement du processus de révision des connaissances. L'*agent diagnostic* est en relation directe avec les *agents connaissance*. Il est également en contact avec le système de résolution de problèmes afin de pouvoir analyser les résultats des instances traitées du problème d'optimisation.

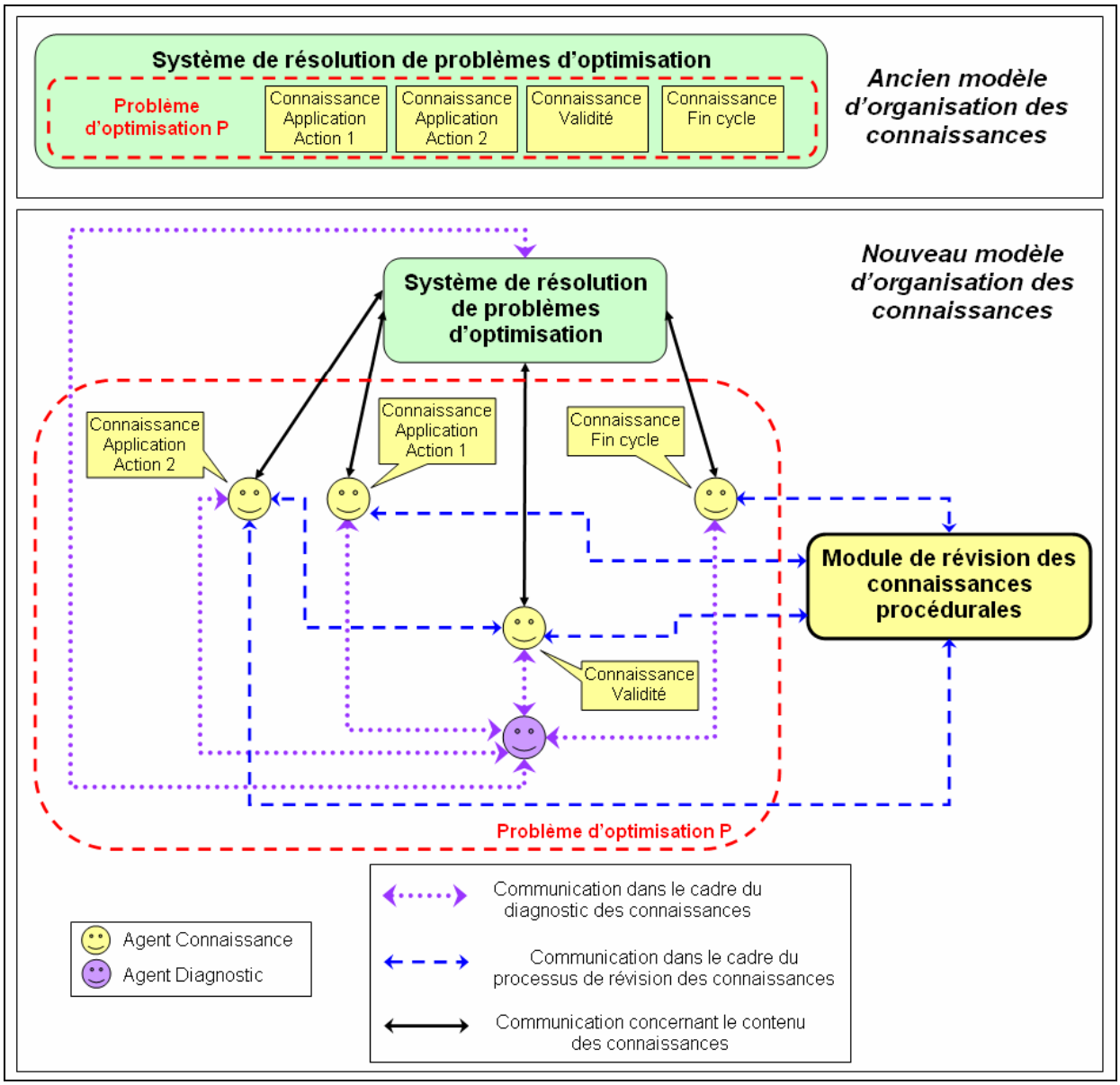


Figure B.12. Nouveau modèle d'organisation des connaissances proposé

La figure B.12 présente le nouveau modèle proposé pour l'organisation des connaissances et présente en particulier les trois types de communication possibles pour les *agents connaissance* : avec le système de résolution de problèmes pour lui fournir, lors de la résolution d'une instance du problème d'optimisation considéré, des informations sur les connaissances que l'*agent connaissance* représente, avec d'autres *agents connaissance* et avec l'*agent diagnostic* dans le cadre du diagnostic des connaissances et avec le module de révision et parfois d'autres *agents connaissance* dans le cadre du processus de révision des connaissances.

Comme illustré par la figure B.12, la mise en place du nouveau modèle d'organisation des connaissances nécessite, d'une part, d'ajouter de nouveaux éléments (les *agents connaissance* et les *agents diagnostic*) et, d'autre part, de modifier le système de résolution de problèmes afin de transférer les connaissances de celui-ci aux *agents connaissance*.

Nous reviendrons sur le rôle des *agents connaissance* au cours du processus de révision en partie D et sur leur rôle dans le cadre du diagnostic en partie E.

B.V.3 Application dans le cadre du modèle AGENT

B.V.3.1 Formalisations et enrichissements proposés

B.V.3.1.1 Présentation des formalisations et enrichissements proposés

Le modèle AGENT, tel que défini initialement, ne propose pas de formalisme clair pour représenter les connaissances. Nous proposons, en vue de permettre l'application de notre approche de révision, de formaliser certains éléments du modèle.

Nous proposons ainsi de modéliser l'ensemble des mesures utilisées par les agents géographiques et par les contraintes sous la forme d'objets au sens informatique du terme. En effet, de nombreuses mesures sont utilisées par les agents géographiques et par les contraintes (pour le calcul de la satisfaction, celui de choix des actions à appliquer, etc.), mais du fait de l'absence de formalisation, il n'est pas possible d'utiliser ces dernières dans le cadre d'un processus automatique de révision. Notre formalisme permet de lier à chaque agent géographique, ainsi qu'à chaque contrainte, une liste de mesures. Un objet "mesure" peut, en fonction de l'état géométrique de l'agent géographique, calculer sa valeur. A noter qu'une mesure peut aussi bien renvoyer une valeur quantitative qu'une valeur qualitative.

Dans un même souci d'automatisation de la révision des connaissances, nous proposons de modéliser les actions sous la forme d'objets au sens informatique du terme. Chaque contrainte a à sa disposition une liste d'actions qu'elle peut proposer à l'agent géographique en fonction de l'état de celui-ci. Cette modélisation permet de ne plus disposer d'une unique base de règles d'application des actions par contrainte, mais d'avoir une base de règles par action. En effet, avec le modèle AGENT classique, chaque contrainte dispose d'une base de règles chargée de définir, en fonction de l'état de l'agent, quelles actions doivent être appliquées et avec quel poids. Notre nouvelle modélisation permet de définir une base de règles par action. Ainsi, chaque action dispose d'une base de règles définissant si l'action doit ou non être appliquée et avec quel poids. Le principal intérêt de disposer d'une base de règles par action est d'améliorer la modularité du système de généralisation. Ainsi, une action peut très simplement être ajoutée ou supprimée du système sans qu'il soit nécessaire de retoucher une quelconque base de règles.

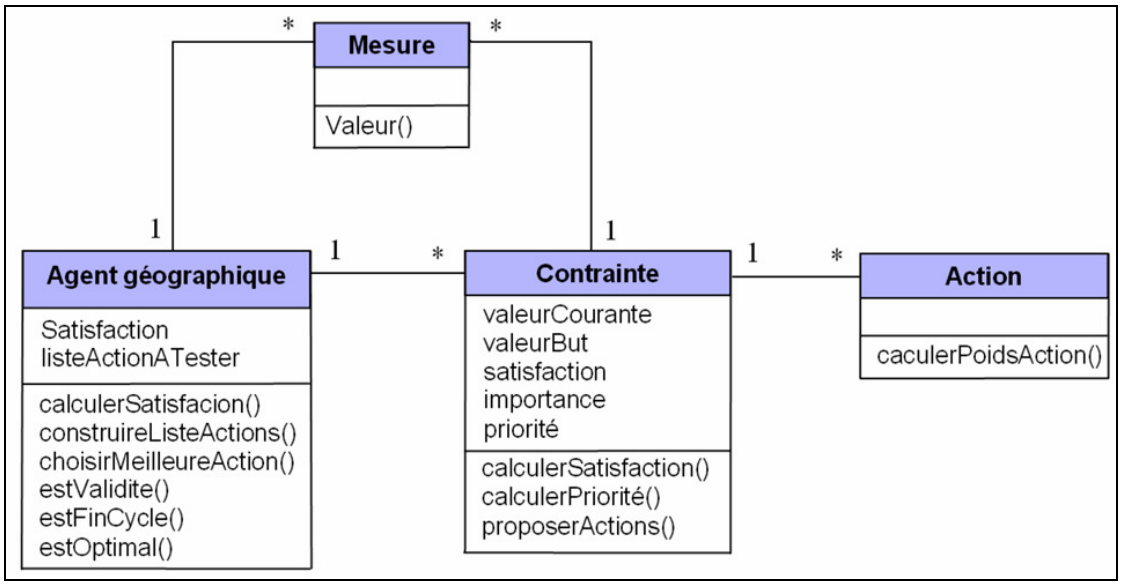


Figure B.13. Diagramme de classe proposé pour le modèle AGENT

La figure B.13 donne le diagramme de classes tenant compte des mesures et des actions que nous proposons.

En plus de cette nouvelle formalisation, nous proposons d'enrichir le modèle AGENT en vue de donner aux utilisateurs un plus grand contrôle sur la stratégie d'exploration employée.

Nous proposons pour cela d'enrichir le cycle d'actions par des critères de fin de cycle et de validité des états plus riches et par un test d'optimalité des états.

Le critère de fin de cycle de base est que le cycle d'actions se termine lorsqu'un état parfait est trouvé ou lorsque tous les états possibles ont déjà été visités. Nous proposons d'enrichir ce critère, par un second critère dépendant de l'état courant de l'agent géographique ainsi que de son état initial. En effet, certains agents géographiques, compte tenu de leur état initial, ne pourront jamais être totalement satisfaits. Il ne sert donc à rien de continuer à développer l'arbre d'états un fois un état satisfaisant trouvé.

Concernant le critère de validité, nous proposons de ne plus considérer qu'un seul critère possible, mais un ensemble de critères possibles (dont celui implémenté dans Clarity™ présenté partie A.IV.4). Chaque type d'agent géographique se voit attribuer l'un de ces critères.

Nous définissons par état optimal, un état *e* dont la qualité (la satisfaction) est supérieure ou égale à celle de tous les états appartenant au sous-arbre dont la racine est l'état *e*. Un état optimal désigne donc, un état qu'il n'est pas possible d'améliorer compte tenu des actions disponibles. Lorsqu'un état optimal est rencontré, l'agent géographique retourne à son état précédent. A signaler que, contrairement au critère de fin de cycle, rencontrer un état optimal ne met pas fin au cycle d'actions, cela ne fait que stopper l'exploration de la branche de l'arbre d'états. Concernant la différence entre le critère d'optimalité et celui de validité, si les effets sont assez similaires (arrêt de l'exploration dans la branche concernée), les causes, elles, sont très différentes : un état est invalide car l'état et le sous-arbre résultant de l'exploration à partir de cet état, semblent peu prometteurs alors qu'un état est optimal

lorsqu'il ne peut plus être amélioré. Contrairement à un état invalide, un état optimal pourra être choisi comme meilleur état trouvé. A noter que ce critère d'optimalité n'est pas lié au modèle AGENT mais peut être utilisé pour tout autre modèle.

La figure B.14 illustre le cycle d'actions enrichi que nous avons proposé.

Dans le cadre de ce travail de thèse, nous désignerons par « modèle AGENT », le modèle enrichi présenté dans cette partie B.V.3.1.

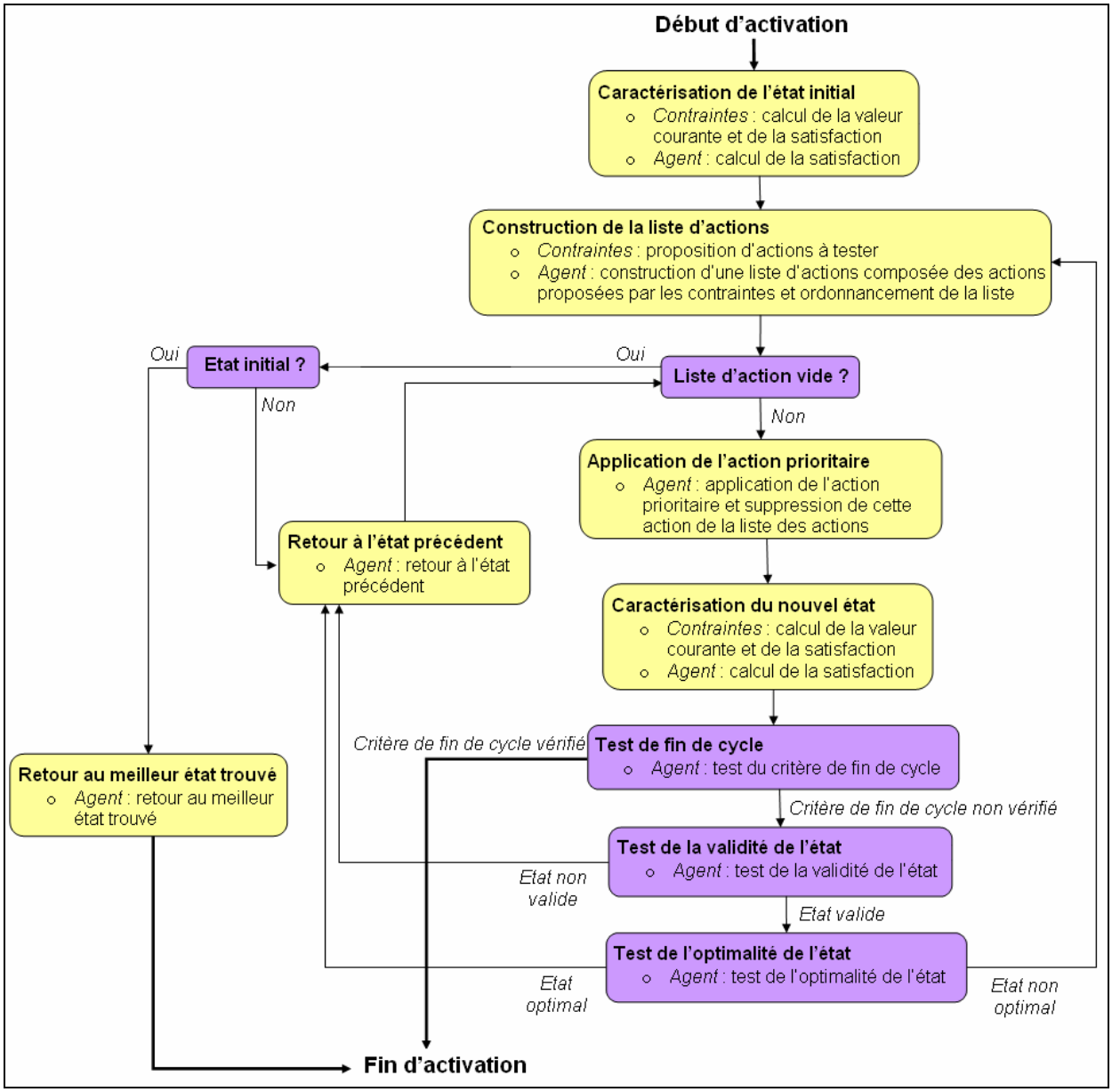


Figure B.14. Cycle d'actions proposé pour le modèle AGENT

B.V.3.1.2 Connaissances procédurales du modèle AGENT enrichi

Nous avons proposé dans la partie B.V.3.1.1, des enrichissements pour le modèle AGENT. Nous récapitulons maintenant les différentes connaissances qui interviennent dans notre version enrichie du modèle AGENT.

Nous notons $Mesures_{agent}(e)$ le vecteur des valeurs prises, pour un état *e* donné, par l'ensemble des mesures définies pour un agent géographique *agent*.

Nous notons de la même manière $Mesures_{contrainte}(e)$ le vecteur des valeurs prises, pour un état *e* donné, par l'ensemble des mesures définies pour une contrainte *contrainte*.

Nous notons $Satisfaction_{Ctr}(e)$ la valeur de la satisfaction de la contrainte *ctr*, pour un état *e* donné.

Nous notons enfin *Condition(val)* une condition de type booléen portant sur l'ensemble des valeurs contenues dans le vecteur de valeurs *val*.
Le modèle tel que nous l'avons enrichi comporte 6 types de connaissances qui sont liées soit à l'agent géographique, soit à une contrainte, soit à une action :

- *Connaissances liées à l'agent géographique* :
  - **Critère d'optimalité** : cette connaissance est représentée sous la forme d'une base de règles définissant si un état (caractérisé par l'ensemble des mesures liées à l'agent géographique) doit être considéré comme optimal ou non.

    Les règles sont du type :
    Si (Condition(Mesures$_{agent}$(état courant)) est vraie
    Alors l'état est optimal

  - **Critère de fin de cycle** : cette connaissance est représentée sous la forme d'une base de règles définissant si le cycle d'actions doit s'arrêter ou non, pour un état donné (caractérisé par l'ensemble des mesures liées à l'agent géographique) et compte tenu de son état initial (caractérisé par le même ensemble de mesures).

    Les règles sont du type :
    Si (Condition(Mesures$_{agent}$(état courant), Mesures$_{agent}$(état initial)) est vraie
    Alors le cycle d'actions s'arrête

  - **Critère de validité** : Nous avons prédéfini un ensemble de critères de validité. Chaque type d'agent géographique se voit attribuer l'un de ces critères. Les critères définis sont composés de sous-critères qui portent, soit sur une comparaison du nouvel état avec l'état précédent, soit sur une comparaison du nouvel état avec l'ensemble des états déjà visités.

    Nous définissons deux sous-critères basés sur une comparaison avec l'état précédent :
    - **CEPAG** (**C**ritère par rapport à l'**E**tat **P**récédent et basé sur l'**A**mélioration de la satisfaction **G**lobale de l'agent) : Etat valide si la satisfaction globale du nouvel état de l'agent géographique est supérieure à celle de l'état précédent
    - **CEPAC** (**C**ritère par rapport à l'**E**tat **P**récédent et basé sur l'**A**mélioration de la satisfaction d'une **C**ontrainte de l'agent) : Etat valide si la satisfaction de la contrainte qui a proposé l'action ayant mené à ce nouvel état a augmenté

Nous définissons deux sous-critères basés sur une comparaison avec l'ensemble des états déjà parcourus :

- **CEENS** (**C**ritère par rapport à l'**E**nsemble des **E**tats parcourus et basé sur la **N**on **S**imilarité de l'état) : Etat valide s'il n'est pas similaire à un état déjà rencontré en termes de satisfaction des contraintes.
- **CEEAP** (**C**ritère par rapport à l'**E**nsemble des **E**tats parcourus et basé sur l'**A**mélioration **P**artielle de l'état) : Etat valide s'il y a *amélioration partielle* par rapport à chacun des états déjà parcourus. Une *amélioration partielle* signifie qu'au moins l'une des contraintes a vu sa satisfaction progresser. Soit un agent géographique avec deux contraintes C_1 et C_2 et un nouvel état ayant pour valeurs respectives de satisfactions pour ses contraintes 4 et 7. Si pour l'un des états déjà parcourus, ces mêmes contraintes ont pour satisfaction 5 et 7, le nouvel état ne sera pas valide. En effet, aucune des deux satisfactions de l'état courant n'est supérieure à celles correspondantes de l'état déjà visité. Par contre, il y a *amélioration partielle* par rapport à un état où C_1 et C_2 ont respectivement pour valeurs 6 et 6, car C_2 a vu sa satisfaction progresser.

Nous proposons de définir le domaine de définition de la connaissance relative à la validité des états par l'ensemble suivant de critères de validité :

- **CEPAC**
- **CEPAC** ET **CEPAG**
- **CEPAC** ET **CEENS**
- **CEPAC** ET **CEEAP**
- **CEPAC** ET **CEPAG** ET **CEENS**
- **CEPAC** ET **CEPAG** ET **CEEA**

Le critère de validité implémenté dans *Clarity*™ et présenté en partie A.IV.4 est le critère **CEPAC** ET **CEEAP**.

Nous pouvons remarquer que le critère **CEPAC** ne permet pas de garantir la convergence théorique du système. De façon à remédier à ce problème, nous proposons de limiter le nombre maximal d'utilisations de chaque action par branche de l'arbre d'états. Nous notons ce nombre maximal d'utilisations ACT_MAX. Ainsi, si l'obtention d'un état a nécessité ACT_MAX applications d'une action, celle-ci ne sera plus proposée pour cet état et pour ses descendants. ACT_MAX est pris suffisamment élevé pour n'avoir que très peu de chances d'intervenir en pratique. En effet, le critère **CEPAC**, qui a une réelle pertinence d'un point de vue cartographique, s'est révélé suffisant pour tous les objets géographiques qui ont été testés durant ce travail de thèse. A titre indicatif, le nombre maximum d'états parcourus est, pour un agent géographique disposant de N actions, de $\frac{(N \times ACT\_MAX)!}{(ACT\_MAX!)^N}$ états.

- *Connaissances liées aux contraintes* :
  - **Priorité des contraintes** : chaque contrainte dispose d'une base de règles définissant, en fonction de sa satisfaction, sa priorité. La base de règles couvre l'ensemble des valeurs possibles de la satisfaction (comprise entre 1 et 10).

Les règles sont du type :

Si (Condition(Satisfaction$_{\text{Contrainte}}$(état Courant)) est vraie
Alors la priorité de la contrainte = val, avec val entier ∈ [1,5]

- *Connaissances liées aux actions* :
  - **Application des actions** : Chaque action dispose d'une base de règles déterminant si, pour un état donné de l'agent géographique (caractérisé par l'ensemble des mesures liées à la contrainte qui propose l'action), l'action doit être appliquée ou non et avec quel poids.

    Les règles sont du type :
    Si (Condition(Mesures$_{\text{contrainte proposant l'action}}$(état courant)) est vraie
    Alors l'action est proposée avec un poids = val,
    avec val entier ∈ [1,5]

  - **Restrictions sur le nombre d'utilisations des actions** : il est possible de limiter le nombre d'utilisations par branche de l'arbre d'états de chaque action. Ainsi, si l'on sait qu'il ne sert à rien d'appliquer plusieurs fois une même action sur un agent géographique, il est possible de définir un nombre maximal d'utilisations pour cette action.

Il est nécessaire de définir chacune de ces connaissances pour chaque classe d'agents géographiques. La figure B.15 donne un exemple de jeu de connaissances défini pour les agents *bâtiments*.

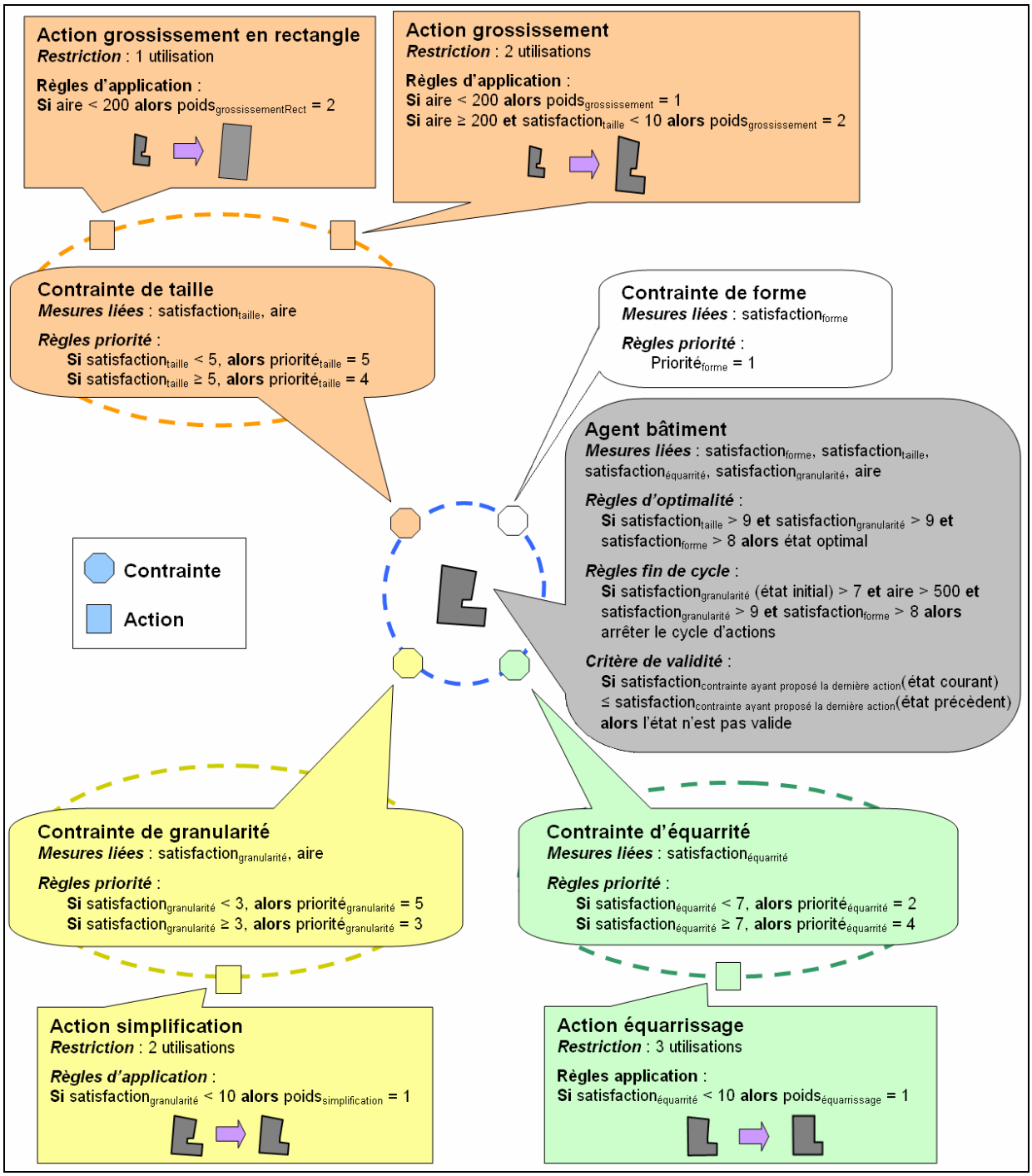


Figure B.15. Exemple de jeu de connaissances pour les agents bâtiment

B.V.3.2 Application de l'approche générale pour le modèle AGENT

Notre approche générale se base sur la dissociation entre le diagnostic des connaissances et leur révision hors ligne.

Comme nous l'avons énoncé dans le cas général, le processus général de révision hors ligne est décomposé en deux phases :

- **Phase d'exploration** : cette phase consiste à généraliser un grand nombre d'objets géographiques et à en tirer des traces. Les objets géographiques seront choisis dans la zone géographique d'étude.
- **Phase d'analyse** : cette phase a pour but d'utiliser les traces obtenues pendant la phase précédente, afin de réviser les connaissances procédurales du système de résolution de problèmes. Dans le cadre du modèle AGENT où six types de connaissances sont représentés, la phase d'analyse consiste à déclencher, pour chaque type de connaissances, un processus permettant de réviser les connaissances de ce type. Il y aura donc un processus de révision lancé pour la révision des règles de priorité des contraintes, un pour la révision des règles d'application des actions, un pour les restrictions d'utilisation des actions, un pour le critère de validité, un pour le critère d'optimalité et un dernier pour le critère de fin de cycle.

B.V.3.3 Agents connaissance pour le modèle AGENT

Nous avons présenté en partie B.V.2, un nouveau modèle d'organisation des connaissances et plus particulièrement les *agents connaissance*. Dans le cadre du modèle AGENT, six types *d'agents connaissance* sont définis : l'*agent connaissance priorité contrainte*, *l'agent connaissance poids action*, *l'agent connaissance restriction action*, *l'agent connaissance validité*, *l'agent connaissance optimalité* et *l'agent connaissance fin cycle*.
Par rapport au modèle AGENT classique, les connaissances sont externalisées : ce ne sont plus directement les contraintes, les actions ou l'agent géographique qui portent les connaissances, mais des agents *connaissance* liés à eux. Un *agent connaissance* de chaque type (*agent connaissance validité*, *agent connaissance d'application* de l'action *grossissement*, etc.) est instancié par classe d'agents géographiques. Dans le cas où plusieurs contraintes proposent une même action, un *agent connaissance application action* est instancié par contrainte : en effet, les deux actions peuvent avoir des connaissances différentes en fonction de la contrainte qui les propose.

Chacun de ces types *d'agents connaissance* a son propre rôle par rapport aux connaissances qu'il fournit au système de résolution de problèmes.
Ainsi, un agent géographique est lié à 3 *agents connaissance* :

- *Agent connaissance validité* : chargé de dire à l'agent géographique si, en fonction de l'ensemble des états déjà parcourus, son état courant est valide ou non.
- *Agent connaissance optimalité* : chargé de dire à l'agent géographique si son état courant est optimal ou non.
- *Agent connaissance fin cycle* : chargé de dire à l'agent géographique si, compte tenu de son état initial, et au vu de son état courant, celui-ci doit mettre fin à son cycle d'actions ou non.

Chacune des contraintes est liée à *l'agent connaissance* :

- *Agent connaissance priorité contrainte* : chargé de calculer pour la contrainte la priorité de cette dernière en fonction de sa satisfaction.

Enfin chacune des actions est liée à 2 *agents connaissance* :

- *Agent connaissance poids action* : chargé de déterminer, en fonction des valeurs prises par l'ensemble des mesures liées à la contrainte, qui a proposé l'action, si l'action doit ou non être appliquée et si oui, avec quel poids.
- *Agent connaissance restriction action* : chargé de déterminer si l'action à laquelle il est lié doit être proposée ou non en fonction du nombre de fois ou celle-ci a été appliquée dans la branche d'arbre concernée.

Un *agent diagnostic* est également défini par classe d'agents géographiques.

La figure B.16 donne un exemple des *agents connaissance* et de l'*agent diagnostic* définis pour les *agents bâtiment*. On peut constater sur cette figure qu'un agent diagnostic a été défini pour les *agents bâtiment*. Concernant les *agents connaissance*, un *agent connaissance optimalité*, un *agent connaissance fin cycle* et un *agent connaissance validité* ont été définis pour les *agents bâtiment*. De même, pour chacune des contraintes, un *agent connaissance priorité contrainte* est défini. Enfin, un *agent connaissance restrictions action* et un *agent connaissance poids action* sont définis par action. On se retrouve, dans le cas des agents *bâtiment* tels que définis ici, avec 15 *agents connaissance* et 1 *agent diagnostic*. Ces 16 agents sont instanciés une seule fois pour tous les agents *bâtiment*. Par exemple, un seul *agent connaissance optimalité* est instancié pour l'ensemble des agents *bâtiment*. La mise en place de notre modélisation a demandé de définir ces 16 agents, mais également de modifier les classes *Contrainte* et *Agent géographique* afin que ces dernières ne contiennent plus les connaissances procédurales mais fassent appel aux *agents connaissance* lorsque nécessaire.

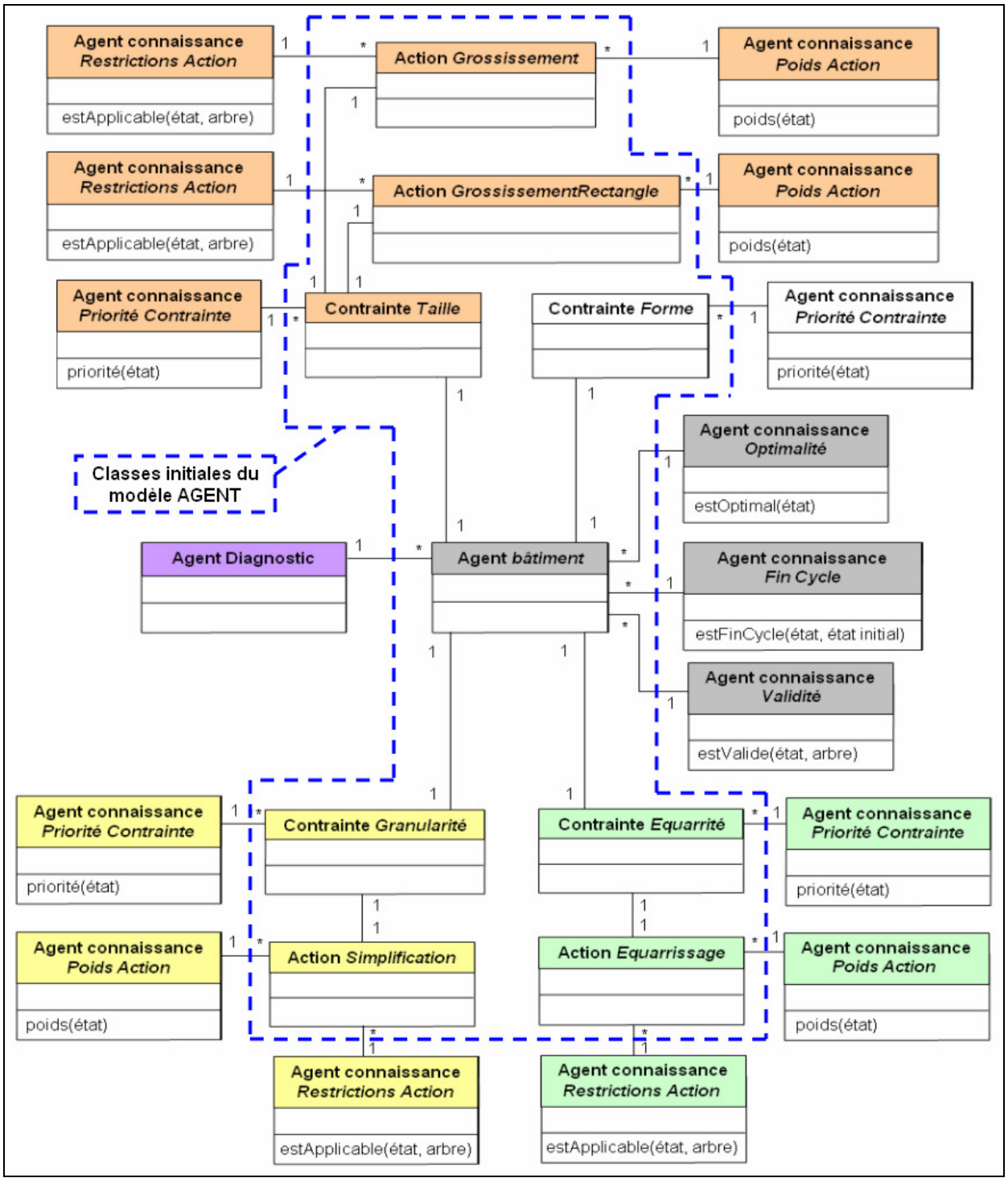


Figure B.16 Agents *connaissance* et *diagnostic* pour les agents *bâtiment*

B.VI Bilan

Nous avons introduit dans ce chapitre le contexte général de l'acquisition et de la révision des connaissances en intelligence artificielle, en particulier dans le domaine de la généralisation de données géographiques. Cette introduction nous a permis de dégager des outils et des problématiques propres à l'acquisition et à la révision des connaissances. Elle nous a également permis de montrer qu'aucun travail existant ne permet de directement répondre à notre sujet de thèse.

Nous avons également proposé, dans ce chapitre, une formalisation du problème de la révision par l'expérience des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états. Cette formalisation nous a servi de base pour poser les problématiques liées à ce type de révision.

Une fois cette problématique établie, nous avons proposé, pour y répondre, une approche de révision ainsi qu'un modèle général pour les connaissances.

L'approche que nous proposons, schématisée en figure B.17, est basée sur le principe d'une révision hors ligne : un *agent diagnostic* est chargé de détecter, lors de l'utilisation normale du système de résolution de problèmes, si les connaissances de ce dernier nécessitent a priori d'être révisées ou non. Dans le cas ou l'*agent diagnostic* détecte un problème au niveau des connaissances, un processus de révision hors ligne est déclenché.

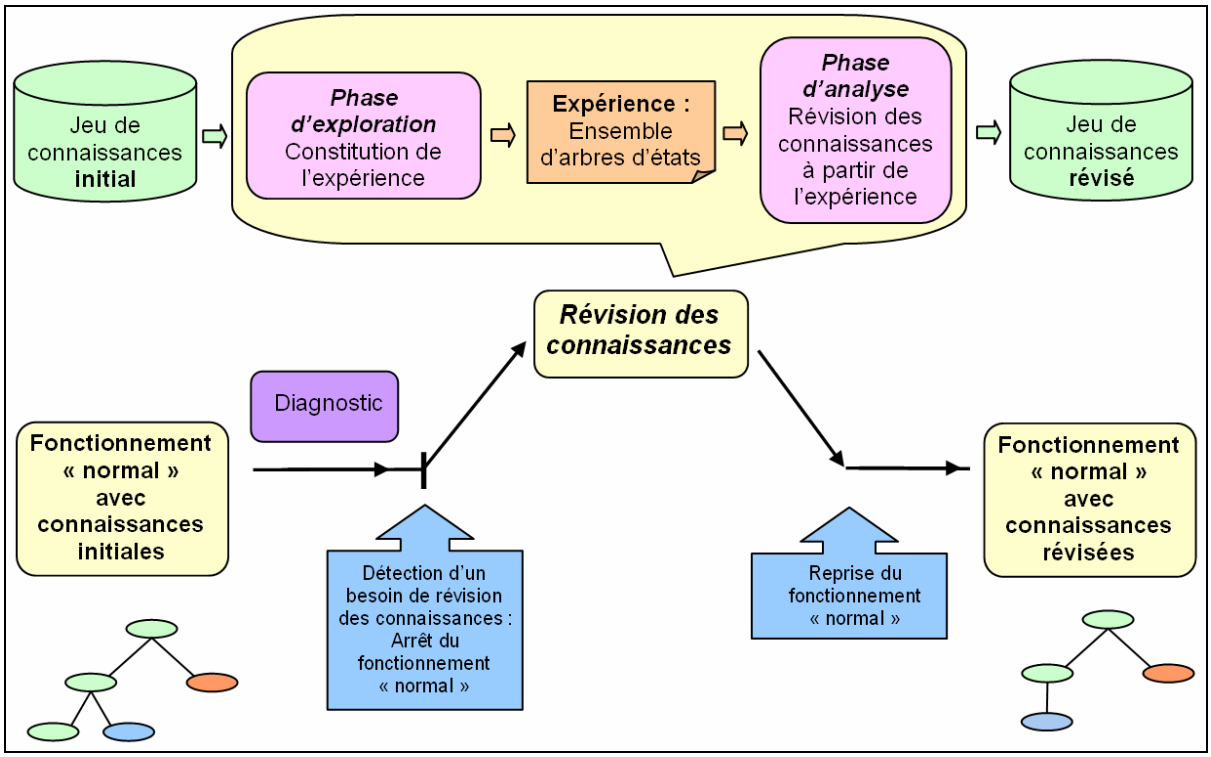


Figure B.17 approche générale détaillée de révision des connaissances

La phase de révision hors ligne est divisée en deux sous phases : la première, la phase *d'exploration*, consiste à engranger de l'expérience, c'est-à-dire produire des traces d'exécution, et la seconde, la phase *d'analyse*, consiste à analyser ces traces pour réviser les connaissances.

Une fois cette approche générale établie, nous avons proposé un modèle général d'organisation des connaissances permettant de la mettre en œuvre. Ce modèle se base sur l'externalisation des connaissances au travers d'agents *connaissance* qui ont pour tâche de gérer les connaissances. Ils sont ainsi chargés de fournir des réponses à des requêtes du système de résolution de problèmes au sujet du contenu des connaissances, de diagnostiquer la qualité du jeu de connaissances ou encore de participer au processus de révision.

Nous avons enfin présenté une formalisation ainsi que des enrichissements du modèle AGENT. En effet, la mise en place de notre approche et de notre modèle dans le cadre de ce modèle a nécessité de formaliser certains éléments du modèle. Les enrichissements proposés ont pour but de donner à l'utilisateur un plus grand contrôle de la stratégie d'exploration.

Nous présentons plus en détail dans le prochain chapitre la phase *d'exploration*, puis nous présentons au chapitre D la phase *d'analyse*.

Chapitre C
Constitution de l'expérience : production de traces d'exécution

C.I Introduction

Nous avons introduit dans le chapitre précédent notre approche générale de révision automatique des connaissances contenues dans un système fonctionnant par exploration informée d'arbres d'états. Celle-ci se base sur l'utilisation de l'expérience passée, matérialisée sous forme de traces qui sont ensuite analysées pour réviser les connaissances du système.

Nous ne cherchons pas à réviser les connaissances en ligne. Notre approche a donc pour vocation de fonctionner hors ligne. Cet aspect « hors ligne » permet, d'une part, au module de révision de choisir lui-même les instances de problèmes qui lui semblent les plus intéressantes à traiter pour constituer l'expérience, et d'autre part, d'utiliser un jeu de connaissances spécifique lors du traçage du processus.

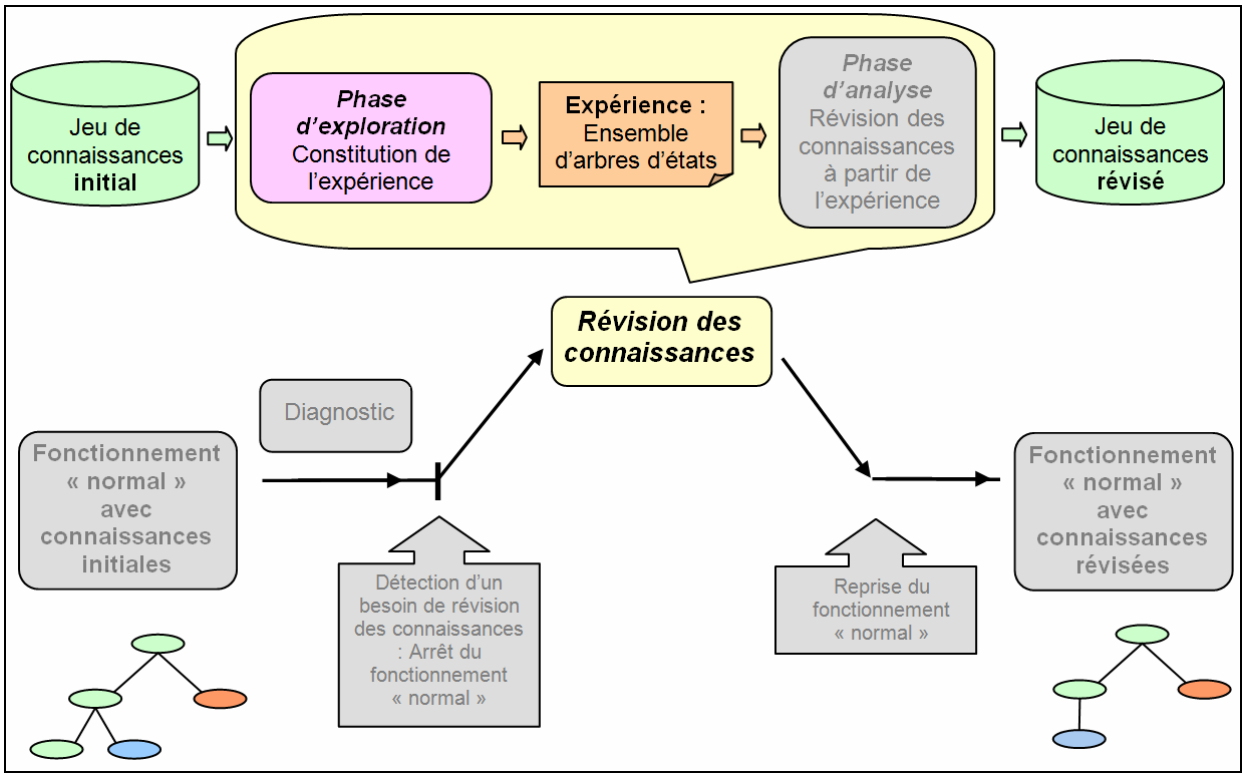


Figure C.1 Phase d'exploration

Nous allons nous intéresser dans ce chapitre à la question de la constitution de l'expérience qui constitue la première phase (la *phase d'exploration*) de notre approche générale (figure C.1). Deux questions se posent : comment choisir l'échantillon d'instances du problème qui sera utilisé par le processus de révision, et à l'aide de quel jeu de connaissances guider le système de résolution de problèmes lors de la production de traces ?

Nous proposerons dans une première partie de nous attarder sur la première question. Nous présenterons une méthode permettant de choisir, parmi un ensemble d'instances du problème d'optimisation considéré, celles qui sont les plus représentatives et donc les plus intéressantes pour le processus de révision.

Nous reviendrons, dans une deuxième partie, sur la question du choix du jeu de connaissances à utiliser lors du traçage du processus. Nous introduirons pour cela, la notion de jeu de connaissances *minimal*. Le système de résolution de problèmes utilisera, en effet, un jeu de connaissances de ce type lors de la production de traces.

Nous nous intéresserons enfin, dans une dernière partie, aux informations complémentaires pouvant être tirées, pour certaines connaissances, de l'analyse locale au niveau de chaque connaissance des arbres d'états. En effet, il est possible, pour les connaissances qui dépendent uniquement d'une situation locale (l'état courant), d'obtenir des informations relatives aux succès et aux échecs rencontrés par celles-ci lors de la résolution d'instances du problème. Ces informations locales, qui prendront la forme de bases d'exemples, seront utilisées pour évaluer la qualité des jeux de mesures à partir desquels sont définies certaines connaissances ainsi que pour en réviser d'autres.

C.II Choix de l'échantillon d'instances du problème utilisé pour la révision des connaissances

C.II.1 Problématique du choix de l'échantillon utilisé pour produire de l'expérience

Nous rappelons que notre approche générale de révision des connaissances se base sur l'analyse de l'expérience afin d'en tirer des informations sur les connaissances du système (cf. B.V.1). Elle nécessite donc de sélectionner parmi l'ensemble des instances du problème disponibles celles qui seront utilisées par le processus de révision.

Comme nous l'avons évoqué en partie B.III.2.1, ce choix est fondamental pour permettre une estimation pertinente de la qualité des connaissances. Un échantillon mal choisi peut entraîner l'obtention de connaissances très performantes pour cet échantillon mais absolument pas adaptées à l'ensemble des instances du problème considéré. On se retrouve face au problème classique en apprentissage artificiel du surapprentissage.

Réussir à choisir des instances du problème réellement adaptées pose plusieurs difficultés.

La première concerne la disponibilité des instances. En effet, dans le cadre de nombreuses applications réelles, seul un nombre restreint d'instances, parmi l'ensemble des instances du problème considéré, sera disponible. Le choix de l'échantillon d'instances devra donc se faire parmi cet ensemble. Un exemple de ce cas de figure concerne la généralisation de données géographiques, où il sera fréquent que la sélection des objets géographiques utilisés pour le processus de révision doive se faire sur une zone restreinte de données pour des raisons pratiques alors que les connaissances sont destinées à être utilisées sur une zone bien plus importante.

Une seconde difficulté provient de l'aspect non prévisible des actions du système. Ainsi, il n'est pas possible de prédire à l'avance quelles informations pourront être tirées de la résolution d'une instance du problème. On ne peut donc pas utiliser les travaux portant sur l'apprentissage actif et proposant des méthodes qui permettent de choisir les exemples d'apprentissage les plus adaptés à une situation courante (Roy & McCallum, 2001 ; Singh et al., 2006).

Nous posons le problème de choix de l'échantillon de révision comme suit :
Soit P_{disp}, un ensemble d'instances disponibles du problème P ($P_{disp} \subset P$). Quels sont les n instances de P à choisir dans P_{disp} pour former l'échantillon de révision P_{rev} ($P_{rev} \subset P_{disp}$) ?

Le nombre d'instances n est forcément inférieur ou égal au nombre d'instances disponibles. Plus ce nombre est élevé, plus l'échantillon a de chances d'être réellement représentatif de l'ensemble des instances du problème concerné, et donc plus les connaissances révisées ont de chances d'être bonnes pour l'ensemble des instances du problème. Le choix de n est souvent contraint par les besoins et les contraintes de l'utilisateur. En effet, très fréquemment, l'utilisateur a des impératifs temporels ne lui permettant pas de faire tourner le processus de révision pendant un temps trop long. Ainsi, si les instances de P sont très complexes à

résoudre et que, par conséquent, chaque résolution prend beaucoup de temps, l'utilisateur peut être contraint de choisir un nombre n très faible. Dans ce contexte, le choix de P_{rev} est important.

C.II.2 Approche proposée de choix de l'échantillon de révision

Plusieurs approches peuvent être adoptées pour le choix de l'échantillon de révision.

Une première approche consiste à demander à un expert du domaine de sélectionner lui-même les instances du problème qui lui semblent les plus « intéressantes ». Ce type d'approche est contraire à la contrainte que nous avons posée de réviser les connaissances sans intervention d'experts.

Une seconde approche consiste à tirer au hasard le nombre souhaité d'instances parmi celles disponibles. Cette approche présente le risque de ne tirer que des instances peu « intéressantes ». En effet, toutes les instances d'un problème n'apportent pas forcément la même quantité d'informations. Ainsi, il peut à la fois y avoir, pour un même problème, des instances extrêmement simples à résoudre, pour lesquelles le système sera toujours très performant quelles que soient ses connaissances, et des instances très complexes à résoudre pour lesquelles des connaissances non adaptées peuvent entraîner de grosses déficiences des performances. Un choix aléatoire peut amener à ne tirer que des instances de la première catégorie. Un échantillon d'instances composé uniquement de ce type d'instances ne permet pas une bonne estimation des performances réelles des jeux de connaissances. Il est donc risqué de se limiter à un choix aléatoire de l'échantillon de révision.

Nous proposons donc une troisième approche basée sur la sélection automatique des instances les plus représentatives de l'ensemble des instances disponibles du problème.

Nous approche se décompose en trois phases :

- Caractérisation des instances disponibles
- Division des instances en familles.
- Choix, dans chacune de ces familles, d'un nombre d'instances proportionnel à la taille du groupe par rapport à l'ensemble des instances disponibles.

C.II.2.1 Phase de caractérisation des instances du problème d'optimisation

Cette phase consiste à construire une base d'objets regroupant les caractéristiques de l'ensemble des instances disponibles du problème considéré. Les instances sont caractérisées à l'aide d'un jeu de mesures décrivant leur état initial.

La définition du jeu de mesures a une grande importance dans le choix de l'échantillon de révision. Il doit donc être suffisamment complet pour décrire avec pertinence les différentes instances du problème mais ne pas être trop riche afin de limiter les risques de redondance entre mesures ainsi que la présence de bruit dans les mesures.

Il existe de nombreuses techniques de filtrage qui peuvent être appliquées afin d'éliminer les mesures peu pertinentes (Mitra et al., 2002 ; Sorg-Madsen et al., 2003 ; Guerif & Bennani, 2007).

C.II.2.2 Phase de définition des familles

Cette phase consiste à diviser les instances du problème en familles à partir de la base d'objets construite durant la phase précédente.

Nous utilisons pour cela des algorithmes de classification automatique non hiérarchique (apprentissage non supervisé) qui vont nous permettre de partitionner l'ensemble des instances disponibles en familles.

Ces algorithmes, très nombreux dans la littérature (Dempster et al., 1977 ; Fisher, 1987), permettent de partitionner un ensemble d'objets décrits par des attributs, en familles disjointes. La plupart de ces algorithmes permettent également d'établir, pour chaque objet, une probabilité d'appartenance à chacune des familles construites.
Un paramètre important de la classification concerne le choix du nombre de familles créées. Si ce dernier est parfois calculé automatiquement au travers de l'algorithme de classification (ex :X-Means (Pelleg & Moore, 2000)), d'autres algorithmes demandent à l'utilisateur de le définir préalablement (ex : K-Means (Duda & Hart, 1973 ; Bishop, 1995)). Il existe, dans la littérature, diverses mesures permettant de définir automatiquement le nombre de familles (Davies & Bouldin, 1979 ; Milligan & Cooper, 1985).

Dans le cadre de notre problème de sélection d'un échantillon représentatif, nous appliquons ces techniques à notre base d'objets de façon à diviser nos instances de problème en familles et à disposer, pour chaque instance, d'une probabilité d'appartenance à chacune de ces familles.

C.II.2.3 Phase de choix des instances du problème

Cette dernière phase consiste à sélectionner, parmi l'ensemble des instances disponibles, celles qui seront utilisées pour la révision.

Nous posons l'hypothèse que l'ensemble des instances disponibles est représentatif de l'ensemble des instances du problème considéré. Nous cherchons donc, durant cette phase, à respecter les proportions d'instances appartenant à chaque famille, afin d'obtenir au final, un échantillon le plus représentatif possible de l'ensemble des instances disponibles et donc, d'après notre hypothèse, de l'ensemble des instances de la classe considérée.

Nous choisissons ainsi, dans chaque famille, les instances les plus représentatives de cette famille, c'est-à-dire, celles qui ont la plus forte probabilité d'en faire partie.

Le nombre n_i d'instances sélectionnées pour une famille i, dépend de l'importance de la famille en termes de quantité d'instances par rapport à l'ensemble des instances disponibles et du nombre d'instances n que l'utilisateur souhaite utiliser pour réviser les connaissances.
n_i se calcule de la manière suivante :

$$n_i = Max\left[E\left(\frac{n * card(famille_i)}{card(P_{disp})} + 0.5 \right), 1 \right]$$

Par exemple, si l'on dispose de 100 instances de P, que l'on souhaite en utiliser 20 pour le processus de révision des connaissances et que la méthode de classification non hiérarchique a divisé les 100 instances en deux familles, la famille 1 constituée de 25 instances et la famille 2

constituée de 75 instances, notre échantillon représentatif sera composé de 5 instances de la famille 1 et de 15 instances de la famille 2.

Notre formule est définie de manière à toujours sélectionner au moins une instance par famille. L'intérêt est de ne pas passer à côté d'une famille d'instances qui, même si elle est peu représentée, peut s'avérer intéressante pour le processus de révision.

Notre formule ne nous garantit pas de sélectionner le nombre exact d'instances souhaité par l'utilisateur. Le fait de prendre au moins une instance par famille et d'utiliser la partie entière peut entraîner le choix d'un nombre d'instances légèrement différent du nombre souhaité. On peut borner cette différence :

$$n - E(\frac{nb\ de\ familles}{3}) \leq card(P_{rev}) \leq n + nb\ de\ familles$$

C.II.3 Application de l'approche pour le système AGENT

Dans le cadre du modèle AGENT, le choix de l'échantillon de révision consiste à sélectionner un ensemble d'objets géographiques de la classe concernée parmi ceux disponibles.

Nous avons introduit, dans le cadre de la formalisation que nous avons proposée pour le modèle AGENT, la notion de jeu de mesures lié à un agent géographique (cf. B.V.3.1).

Nous proposons donc d'utiliser, pour caractériser les agents géographiques (et plus précisément leur état initial), le jeu de mesures qui est lié à la classe d'agents géographiques considérée. Nous construisons notre base d'objets, à partir de ce jeu de mesures et de l'ensemble des agents géographiques de la classe considérée.

La figure C.2 donne un exemple de construction de base d'objets pour les agents *bâtiment*. Nous pouvons constater sur cet exemple que caractériser l'état d'un agent géographique uniquement par la satisfaction de ses contraintes est souvent insuffisant. En effet, sur cet exemple, le bâtiment de gauche et celui du centre ont exactement les mêmes valeurs de satisfaction pour leurs contraintes, or ces deux bâtiments sont assez différents. L'ajout de mesures supplémentaires a permis de caractériser cette différence.

Il est ainsi souvent nécessaire d'ajouter, en plus des satisfactions des contraintes, des mesures supplémentaires. L'ajout de mesures est particulièrement important dans le cas d'agents géographiques disposant de nombreuses contraintes de « conservation ». Ainsi, dans le cadre des agents *bâtiment*, la contrainte de concavité, qui a pour rôle de vérifier la conservation de la forme, a toujours une valeur de satisfaction de 10 pour l'état initial d'un agent *bâtiment*. Elle n'apporte donc aucune information sur les caractéristiques des états initiaux des agents *bâtiment*.

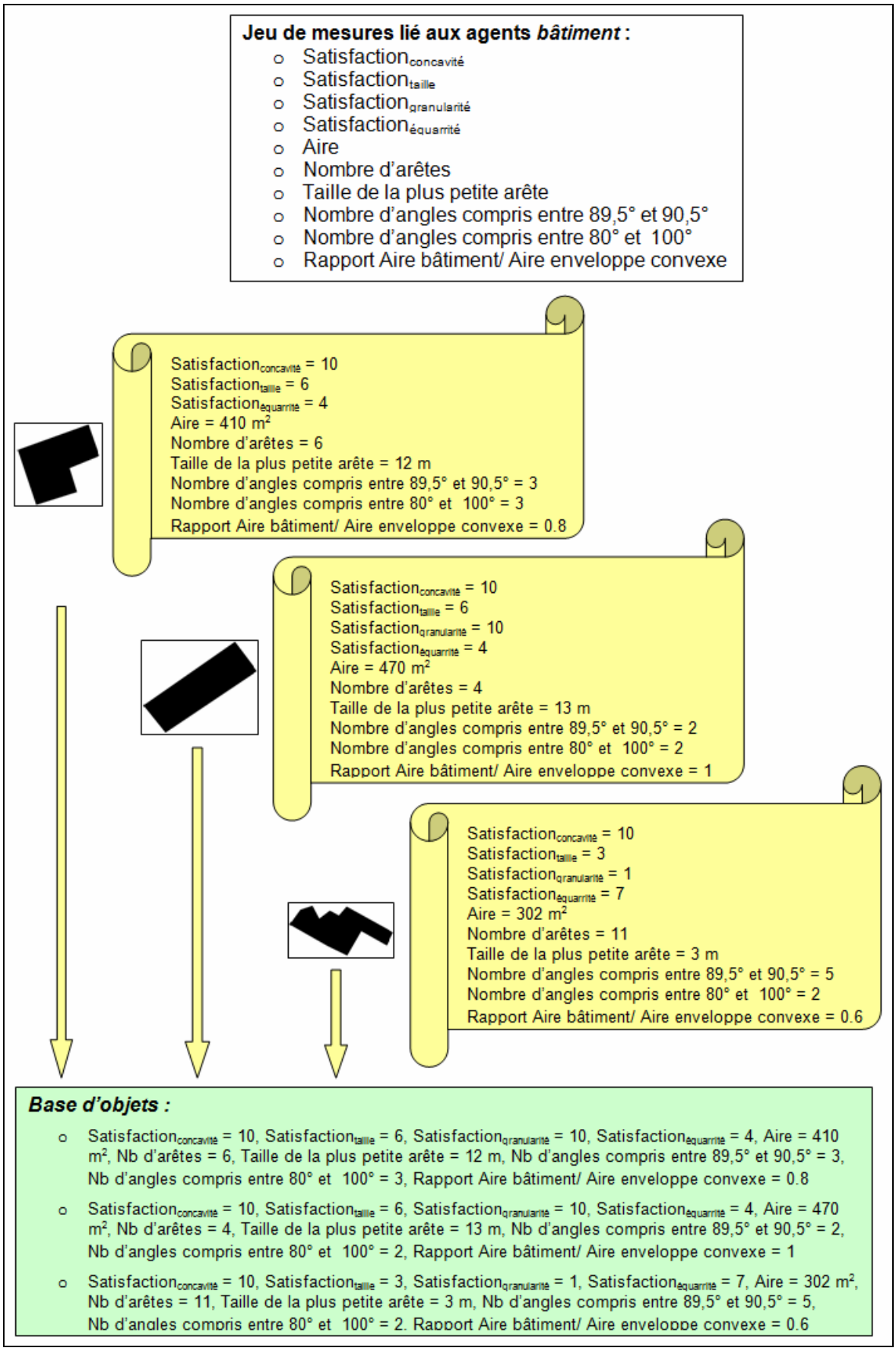


Figure C.2 Exemple de construction d'une base d'objets pour les agents *bâtiment*

C.III Choix du jeu de connaissances utilisé lors de la production de traces

C.III.1 Importance du jeu de connaissances choisi lors de la production de traces

Nous avons évoqué en partie B.IV.1 les problèmes liés à la dépendance des connaissances entre elles. Cette dépendance peut biaiser les informations qui peuvent être tirées de la résolution d'une instance d'un problème.

L'aspect hors-ligne du processus de révision des connaissances permet de faire face à cette difficulté. En effet, il permet de définir des valeurs spécifiques pour les connaissances pour la phase de constitution de l'expérience. Nous utilisons ainsi, durant cette phase d'exploration, un jeu de connaissances dit *minimal* qui permet de construire l'ensemble des états atteignables par l'ensemble des jeux de connaissances possibles. Ainsi, une fois qu'une instance de problème d'optimisation aura été résolue avec ce jeu de connaissances, il deviendra possible de simuler, pour cette instance, tous les jeux de connaissances existants sans avoir à refaire de calculs lourds.

La définition de notre jeu de connaissances *minimal* est basée sur la relation « *d'élagage moins sévère ou égal* ». Nous définissons cette relation ainsi :
Soit P, un problème et K_P, l'ensemble des jeux de connaissances possibles pour ce problème.
Un jeu de connaissance $k \in K_P$ est d'*élagage moins sévère ou égal* que $k' \in K_P$ ($k \leq_{élagage} k'$) si et seulement si :

$$\forall p \in P,\ \forall e \in m_p(S_{k'}),\ e \in m_p(S_k)$$

Ainsi, un jeu de connaissances k est d'*élagage moins sévère ou égal* qu'un autre jeu k', si et seulement si, quelle que soit l'instance $p \in P$ à traiter, tous les états construits pour résoudre cette instance avec k' ont aussi été construits avec k.

Nous pouvons démontrer certaines propriétés mathématiques de la relation $\leq_{élagage}$ pour tout K_P.:

- Réflexivité : Soit $k \in K_P$, $k \leq_{élagage} k$
- Transitivité : Soit $(k, k', k'') \in K_P^3$, $k \leq_{élagage} k' \wedge k' \leq_{élagage} k'' \Leftrightarrow k \leq_{élagage} k''$

Par contre, notre relation d'élagage n'est pas antisymétrique pour tout K_P. En effet, nous pouvons démontrer que, pour certains K_P, deux jeux de connaissances différents peuvent permettre de construire des états identiques :

- $\exists (k, k') \in K_P^2 \ / \ k \leq_{élagage} k' \wedge k' \leq_{élagage} k \wedge k \neq k'$

Le modèle AGENT est un exemple de cadre où l'on peut trouver des K_P où la propriété d'antisymétrie n'est pas vraie. Soit Ag une classe d'agents géographiques. Nous définissons un premier jeu de connaissances K_1 pour Ag tel qu'aucune action ne soit proposée pour aucun

état. Le seul état visité par un agent de la classe *Ag* avec utilisation de K_1 sera son état initial. Nous définissons un second jeu de connaissances K_2 pour *Ag* tel que le critère de fin de cycle entraîne, pour tout état, la fin du cycle d'actions. Pour ce jeu de connaissances aussi, le seul état visité par un agent de la classe *Ag* avec utilisation de K_2 sera son état initial. Nous pouvons donc montrer que $K_1 \leq_{élagage} K_2 \wedge K_2 \leq_{élagage} K_1$, or $K_1 \neq K_2$.

La relation $\leq_{élagage}$ n'est donc pas une relation d'ordre pour tout K_P.

Nous pouvons aussi démontrer qu'il ne sera pas possible, pour tout K_P, de comparer l'ensemble des jeux de connaissances entre eux par la relation $\leq_{élagage}$. La figure C.3 donne des exemples de jeux de connaissances non comparables par la relation $\leq_{élagage}$: le jeu de connaissances présenté dans le cadre 3 n'est pas comparable avec celui présenté dans le cadre 5.

Nous définissons la notion de « *jeu de connaissances minimal* » par :
Un jeu de connaissances $k \in K_P$ est dit *minimal* pour K_P si et seulement si k est d'*élagage moins sévère ou égal* que tous les éléments de K_P :

$$k \in K_P \text{ est minimal pour } K_P \Leftrightarrow \forall k' \in K_P,\ k \leq_{élagage} k'$$

La figure C.3 donne un exemple de jeu de connaissances *minimal*. Dans cet exemple, deux types de connaissances procédurales interviennent :

- Choix des actions à appliquer :
  - Domaine de définition :
    - Aucune action n'est proposée
    - *Action 1* est toujours proposée mais jamais *Action 2*
    - *Action 2* est toujours proposée mais jamais *Action 1*
    - Les deux actions sont toujours proposées.

- Critère de validité :
  - Domaine de définition :
    - Un état est valide lorsque la qualité de cet état est supérieure à la qualité de son prédécesseur
    - Un état est valide lorsque la qualité de celui-ci est supérieure ou égale à la qualité de son prédécesseur.

Nous rappelons que la notion de domaine de définition d'une connaissance a été introduite en partie B.III.1.2.

En combinant ces deux types de connaissances, nous arrivons à un ensemble de huit jeux de connaissances possibles. Le jeu de connaissances *minimal* correspond au jeu de connaissances 8 : les deux actions sont proposées pour tous les états et les états sont valides si leur qualité est supérieure ou égale à celle de leur prédécesseur.

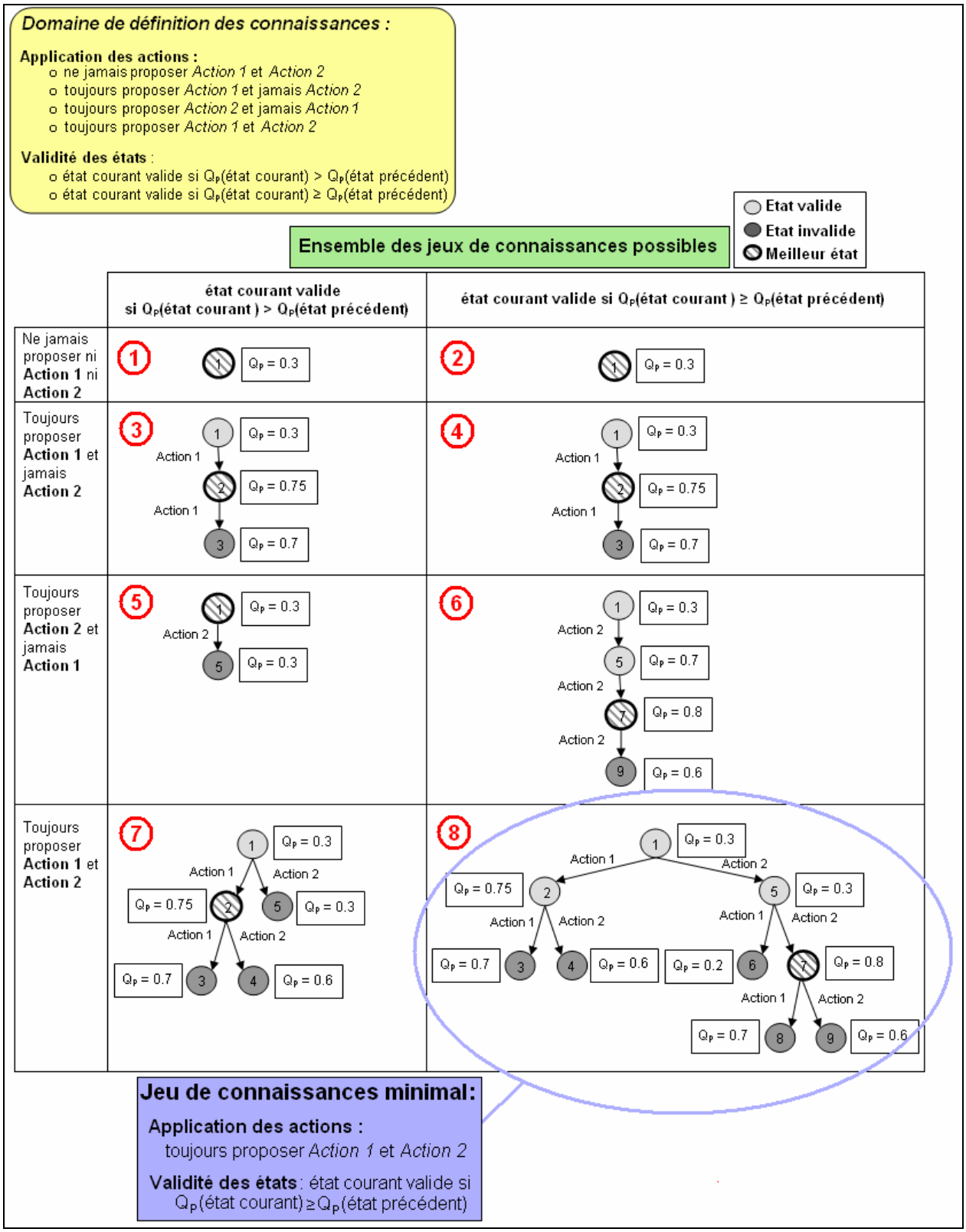


Figure C.3 Exemple de jeu de connaissances *minimal*

Nous rappelons que l'intérêt du jeu de connaissances *minimal* est qu'il permet de construire tous les états qu'il est possible de visiter avec chacun des jeux de connaissances. Une fois une instance de problème d'optimisation résolue avec ce jeu de connaissances, il devient possible

de simuler tous les jeux de connaissances existants sans avoir à refaire de calculs lourds. En effet, tous les états atteignables ayant déjà été construits, il suffit, pour tester un jeu de connaissances, de réarranger ces états en fonction du jeu de connaissances considéré. Par exemple, sur la figure C.3, pour obtenir l'arbre d'états correspondant au jeu de connaissances {toujours appliquer *Action 1* et jamais *Action 2* ; état valide si la qualité de cet état est supérieure à la qualité de son prédécesseur}, il a suffi de réarranger les états 1, 2 et 3 de l'arbre d'états construit avec le jeu de connaissances *minimal*.

Il n'existe pas toujours de jeu de connaissances *minimal* dans un ensemble de jeux de connaissances. En effet, si nous reprenons l'exemple de la figure C.3 et que nous enlevons, comme valeur possible pour la connaissance d'application des actions, la valeur : « toujours proposer Action 1 et Action 2 », il n'est plus possible de définir de jeu de connaissances *minimal*. En effet, aucun des 6 jeux de connaissances n'admet *d'élagage moins sévère ou égal* que l'ensemble des autres. Nous notons K_i le jeu de connaissances n°i de la figure C.3.

- Jeu de connaissances K_1 :
  - $K_1 \leq_{élagage} K_1$
  - $K_1 \leq_{élagage} K_2$
  - K_1 n'admet pas *d'élagage moins sévère ou égal* que K_3, K_4, K_5 et K_6
- Jeu de connaissances K_2 :
  - $K_2 \leq_{élagage} K_1$
  - $K_2 \leq_{élagage} K_2$
  - K_2 n'admet pas *d'élagage moins sévère ou égal* que K_3, K_4, K_5 et K_6
- Jeu de connaissances K_3 :
  - $K_3 \leq_{élagage} K_1$
  - $K_3 \leq_{élagage} K_2$
  - $K_3 \leq_{élagage} K_3$
  - $K_3 \leq_{élagage} K_4$
  - K_3 n'admet pas *d'élagage moins sévère ou égal* que K_5 et K_6
- Jeu de connaissances K_4 :
  - $K_4 \leq_{élagage} K_1$
  - $K_4 \leq_{élagage} K_2$
  - $K_4 \leq_{élagage} K_3$
  - $K_4 \leq_{élagage} K_4$
  - K_4 n'admet pas *d'élagage moins sévère ou égal* que K_5 et K_6
- Jeu de connaissances K_5 :
  - $K_5 \leq_{élagage} K_1$
  - $K_5 \leq_{élagage} K_2$
  - $K_5 \leq_{élagage} K_5$
  - K_5 n'admet pas *d'élagage moins sévère ou égal* que K_3, K_4 et K_6
- Jeu de connaissances K_6 :
  - $K_6 \leq_{élagage} K_1$
  - $K_6 \leq_{élagage} K_2$
  - $K_6 \leq_{élagage} K_5$
  - $K_6 \leq_{élagage} K_6$
  - K_6 n'admet pas *d'élagage moins sévère ou égal* que K_3 et K_4

Notre approche de révision des connaissances est basée sur la résolution, avec un jeu de connaissances *minimal,* d'instances du problème d'optimisation considéré. Il est donc nécessaire, pour l'appliquer, de disposer d'un tel jeu de connaissances. Dans le cas contraire,

il sera nécessaire d'enrichir le domaine de définition des connaissances afin de pouvoir en définir un.

Le jeu de connaissances *minimal* ne sera pas unique pour tout K_P. Ainsi, nous démontrerons en partie C.III.2 qu'il existe plusieurs jeux de connaissances *minimaux* pour le modèle AGENT.

C.III.2 Jeu de connaissances minimal pour le modèle AGENT

Nous définissons comme jeu de connaissances *minimal* pour le modèle AGENT, le jeu de connaissances suivant :

- *Connaissances liées à l'agent géographique* :
  - **Critère d'optimalité** : Aucun état n'est optimal.
  - **Critère de fin de cycle** : Le cycle d'actions s'arrête uniquement lorsque toutes les actions ont été testées pour tous les états de l'arbre. Même lorsqu'un état parfait est atteint, le système continue à explorer les autres branches de l'arbre d'états.
  - **Critère de validité** : le critère de validité attribué est le critère **CEPAC** (cf. B.V.3.1.2) : un état est considéré comme valide si la satisfaction de la contrainte qui a proposé l'action ayant mené à ce nouvel état a augmenté.
    Soit K_{CV} un jeu de connaissances K dont le critère de validité a pour valeur CV. Nous pouvons démontrer que :
    - $K_{CEPAC} \leq_{élagage} K_{CEPAC}$
    - $K_{CEPAC} \leq_{élagage} K_{CEPAC\ ET\ CEPAG}$
    - $K_{CEPAC} \leq_{élagage} K_{CEPAC\ ET\ CEENS}$
    - $K_{CEPAC} \leq_{élagage} K_{CEPAC\ ET\ CEEA}$
    - $K_{CEPAC} \leq_{élagage} K_{CEPAC\ ET\ CEPAG\ ET\ CEENS}$
    - $K_{CEPAC} \leq_{élagage} K_{CEPAC\ ET\ CEPAG\ ET\ CEEA}$

- *Connaissances liées aux contraintes* :
  - **Priorité des contraintes** : ces connaissances n'interviennent pas dans l'élagage des arbres d'états, mais seulement dans l'ordre d'application des actions. Il est donc possible de choisir n'importe quelles connaissances de priorité pour le jeu de connaissances *minimal*. Comme nous l'avons évoqué en partie C.III.1, il n'existe donc pas qu'un seul jeu de connaissances *minimal* pour le modèle AGENT.

- *Connaissances liées aux actions* :
  - **Application des actions** : Toutes les contraintes non parfaitement satisfaites proposent, pour chaque état valide, l'ensemble des actions qu'elles peuvent proposer (cf. B.V.3.1.1). Il est inutile pour une contrainte parfaitement satisfaite de proposer une action, car, quel que soit le résultat, sa satisfaction ne pourra pas être améliorée et donc le nouvel état ne sera jamais valide.
  - **Restrictions sur le nombre d'utilisations des actions** : aucune restriction n'est définie. Toutes les actions peuvent être appliquées autant de fois que nécessaire.

La figure C.4 donne un exemple d'un agent *bâtiment* extrêmement simple généralisé avec un jeu de connaissances *minimal*. On retrouve bien dans cet exemple, un arbre d'états complet où le cycle d'actions a continué l'exploration malgré l'obtention d'états parfaits (les états 3, 5, 8 et 9).

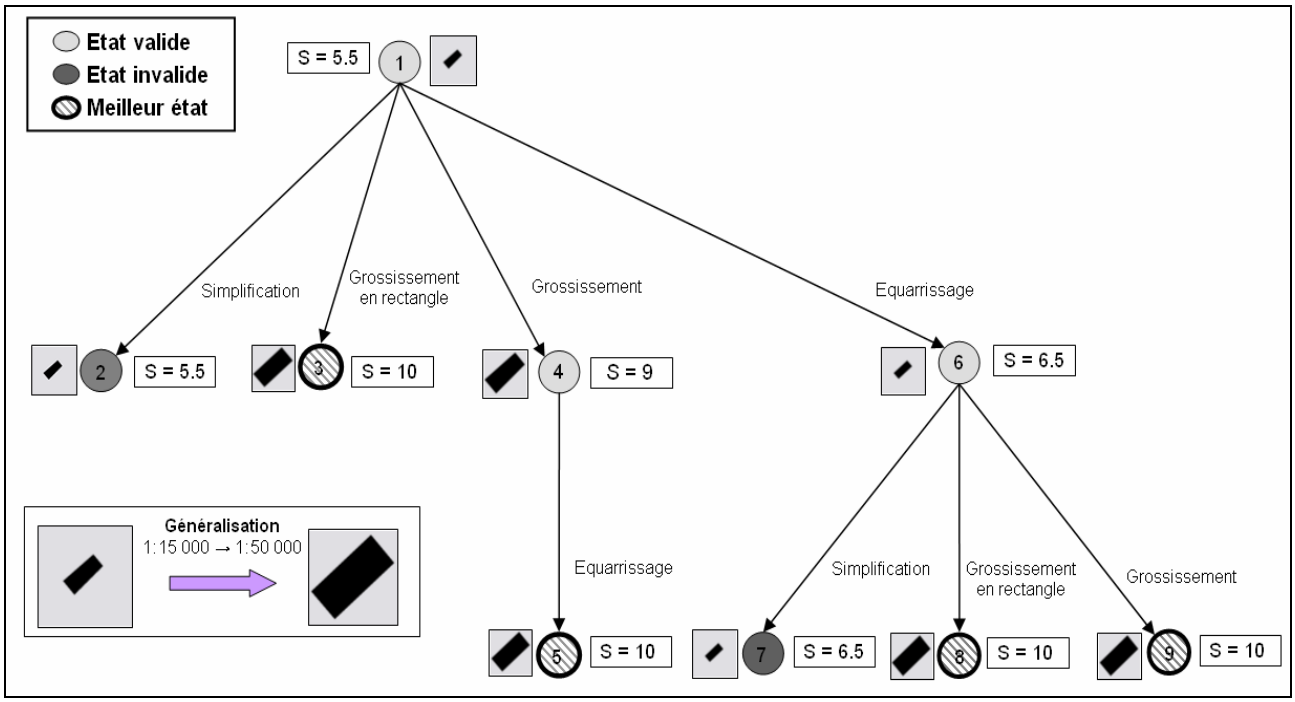


Figure C.4 Exemple de généralisation obtenue pour un agent *bâtiment* avec un jeu de connaissances *minimal*

C.IV Construction des bases d'exemples

C.IV.1 Introduction

Nous avons introduit dans les parties précédentes les problématiques liées à la production de l'expérience. Cette expérience prend la forme d'un ensemble d'arbres d'états où chaque arbre représente l'ensemble des états pouvant être construits lors de la résolution de l'une des instances de notre échantillon représentatif. Il est possible, pour certaines connaissances, de tirer des informations complémentaires de ces arbres d'états en analysant localement les succès et les échecs rencontrés par chaque connaissance. Nous nous intéressons dans cette partie C.IV à cette analyse locale.

Comme nous l'avons évoqué en partie C.I, cette analyse ne peut être menée que pour les connaissances qui dépendent de l'état courant de l'entité considérée. En effet, certaines connaissances dépendent uniquement d'une situation locale (l'état courant) pour prendre une décision, alors que d'autres dépendent d'une situation globale (l'ensemble des états parcourus). Le fait de dépendre uniquement d'un état courant et non de l'ensemble des états parcourus permet d'analyser directement, au niveau de cet état, quelle décision prendre. Ainsi, quelle que soit la position de l'état dans l'arbre, les décisions prises à partir de cet état pourront être analysées de la même façon. Il ne sera pas possible de mener ce même type d'analyse pour les connaissances dépendantes de la situation globale car l'analyse de telles connaissances nécessite de prendre en considération l'ensemble des états déjà parcourus avant d'arriver à cet état. Des exemples de connaissances dépendant uniquement d'une situation locale sont les connaissances relatives à la priorité des contraintes du modèle AGENT (cf. B.V.3.1.2).

Nous formalisons ces analyses locales sous la forme de bases d'exemples. Nous proposons de construire une base d'exemples par connaissance ainsi qu'une base d'exemples supplémentaire pour le cas de plusieurs connaissances directement liées entre elles. En effet, certaines connaissances ne peuvent être analysées indépendamment d'autres. C'est le cas des connaissances relatives à la priorité des contraintes du modèle AGENT. Si l'expression des règles de priorité de chaque contrainte se fait de façon totalement indépendante (chaque contrainte dispose d'une base de règles qui ne dépend que de sa propre satisfaction), le résultat obtenu (la priorité) ne peut être analysé que s'il est comparé aux priorités des autres contraintes de l'agent géographique. Par exemple, pour un agent géographique qui a deux contraintes C_1 et C_2, savoir que, pour un état donné, la priorité de C_1 vaut 5 n'a en soi pas de sens si on ne sait pas que pour le même état, la priorité de C_2 est égale à 3. La base d'exemples supplémentaire commune à un ensemble de connaissances, a pour objectif d'aider à l'analyse des interactions existantes entres les différentes connaissances. Nous reviendrons sur la forme des bases d'exemples ainsi que sur leur construction en partie C.IV.2.

Comme évoqué précédemment, les connaissances qui nous intéressent dans cette partie C.IV dépendent uniquement de l'état courant. Elles s'appuient sur un jeu de mesures pour caractériser les différents états pouvant être pris par l'entité considérée. Or ce jeu de mesures peut s'avérer inadapté pour caractériser avec pertinence les états pouvant être pris par l'entité. Nous proposerons donc, en partie C.IV.3, une approche visant à évaluer, à partir des bases d'exemples construites, la qualité d'un jeu de mesures lié à une connaissance. A des fins de

formalisation, nous considérons qu'un jeu de mesures est défini par connaissance. En pratique, un même jeu de mesures pourra être commun à deux connaissances.

C.IV.2 Construction des bases d'exemples

C.IV.2.1 Forme des bases d'exemples

Nous nous intéressons dans cette partie C.IV.2.1 à la forme des bases d'exemples.

Les exemples de ces bases sont représentés sous le formalisme attributs/valeur. Un exemple est ainsi composé d'un état ainsi que d'une conclusion pour cet état.

L'état correspond à un état de l'entité considérée (un état d'un arbre d'états) caractérisé par un ensemble d'attributs qui dépend de la nature de la connaissance. Dans le cas d'une base d'exemples construite pour une unique connaissance, l'ensemble d'attributs correspond au jeu de mesures lié à la connaissance. Dans le cas d'une base d'exemples construite pour plusieurs connaissances interdépendantes, l'ensemble d'attributs correspond à l'union de l'ensemble des jeux de mesures liés à chacune des connaissances.

La conclusion correspond à une étiquette affectée à l'état pour la connaissance considérée. Un exemple de conclusion peut être, pour une connaissance d'application d'une action, que l'application de cette action pour l'état concerné par l'exemple a entraîné un « échec » dans le cadre du traitement des instances de l'échantillon de révision. La figure C.5 présente le diagramme de classes des bases d'exemples et la figure C.6 un exemple de base d'exemples.

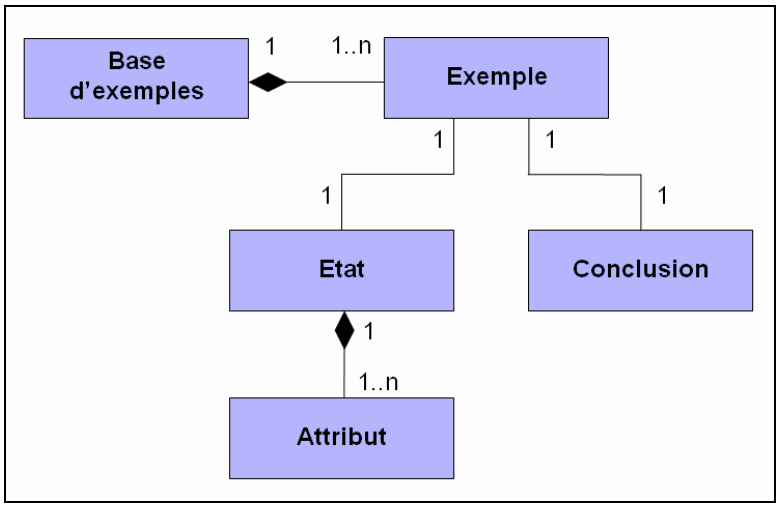


Figure C.5 Diagramme de classes des bases d'exemples

Jeu de mesures JM_K associé à K : $\{M_1, M_2, M_3, M_4, M_5\}$

Conclusions possibles pour K : {« succès », « échec »}

Base d'exemples construite pour K :

Etat					Conclusion
M_1	M_2	M_3	M_4	M_5	
1	0.1	122	3.2	2.5	« succès »
1	0.1	1457	18	4.5	« succès »
2	0.1	1478	-1	25.5	« succès »
2	0.1	5547	14	4.5	« échec »
3	0.1	1800	20	2.5	« échec »
5	0.1	7744	-0.56	14.5	« succès »
10	0.1	4415	16	24.5	« succès »
20	0.1	478	17	14.5	« échec »

Figure C.6 Exemple de base d'exemples

Nous formalisons de la façon suivante une base d'exemples :
Soit un problème d'optimisation P, un système S et une connaissance K utilisée par S pour résoudre les instances de P. Soit M_K, le jeu de mesures lié à K et C_K, l'ensemble des conclusions possibles pour les éléments de K.

Soit $B_K = \{ex_i\}_{i\in[1,N]}$, une base de N exemples construite après analyse des arbres d'états obtenus durant la phase d'exploration. Les ex_i représentent les N exemples de B_K.

Chaque exemple ex_i est composé d'un état e_i et d'une conclusion c_i : $ex_i = (e_i, c_i)$.
Un état e_i est caractérisé par les valeurs prises par l'ensemble des mesures de M_K :

$$e_i = \{Valeur(m_j)\}_{m_j \in M_K}$$

Une conclusion est caractérisée par sa valeur prise dans C_K.

Nous nous intéressons dans la partie suivante (C.IV.2.2) au problème de la construction des bases d'exemples à partir des arbres d'états.

C.IV.2.2 Construction des bases d'exemples à partir des arbres d'états

Le processus de construction d'une base d'exemples est fondé sur l'analyse des arbres d'états obtenus durant la *phase d'exploration*. Le processus est dépendant du type de connaissances concernées. En effet, en fonction du type de la connaissance pour laquelle la base d'exemples est construite, la construction de cette base peut différer.
Par exemple, pour le modèle AGENT, le processus de construction de la base d'exemples liée à la connaissance sur la fin de cycle et celui de construction de la base d'exemples liée à la connaissance sur l'application d'une action sont très différents. En effet, le fait que pour un état donné l'agent géographique doive ou non continuer à explorer l'arbre d'états n'est pas forcément lié au fait qu'il doive ou non appliquer une action donnée. Pour certains états, l'agent géographique devra continuer à explorer l'arbre d'états mais ne devra pas appliquer certaines actions.

Le processus de construction d'une base d'exemples peut donc différer d'un type de connaissance à un autre. Néanmoins, la plupart des processus de construction s'appuient sur l'analyse des *meilleurs chemins* des arbres d'états.

Nous définissons la notion de *meilleur chemin* comme la séquence *sq* d'au moins deux états ayant pour état initial un état e_i non terminal de l'arbre et pour état final le meilleur état du sous-arbre ayant pour racine l'état e_i et tel qu'il n'y ait pas d'autre meilleur chemin qui contienne, tous les états de *sq* dans sa séquence d'états.

Afin de formaliser la notion de meilleur chemin, nous posons les notations suivantes :
Soit *AE*, un arbre d'états. Nous notons MC_{AE}, l'ensemble des meilleurs chemins de *AE*.

Nous notons $[e_i,\ldots,e_f]_{AE}$ la séquence de $card([e_i,\ldots,e_f]_{AE})$ états de *AE* ayant pour état initial, l'état e_i et pour état final, l'état e_f.

Nous notons $Meilleur\_etat_{AE}(e)$, l'ensemble des meilleurs états du sous-arbre de *AE* ayant pour état initial l'état *e* (il peut y avoir plusieurs états avec la même valeur de Q_P).

Nous définissons enfin la notion d'inclusion entre deux séquences d'états :
Si l'ensemble des états d'une séquence $[e_i,\ldots,e_f]_{AE}$ est inclus dans la séquence $[e_i',\ldots,e_f']_{AE}$, alors $[e_i,\ldots,e_f]_{AE}$ est incluse dans $[e_i',\ldots,e_f']_{AE}$ et nous écrirons $[e_i,\ldots,e_f]_{AE} \subset [e_i',\ldots,e_f']_{AE}$.
Si au moins un état de la séquence $[e_i,\ldots,e_f]_{AE}$ n'est pas inclus dans la séquence $[e_i',\ldots,e_f']_{AE}$, alors $[e_i,\ldots,e_f]_{AE}$ n'est pas inclus dans $[e_i',\ldots,e_f']_{AE}$ et nous écrirons $[e_i,\ldots,e_f]_{AE} \not\subset [e_i',\ldots,e_f']_{AE}$

La notion de meilleur chemin pour un arbre d'états *AE* peut donc être formalisée par :

$$[e_i,\ldots,e_f]_{AE} \in MC_{AE} \Leftrightarrow card([e_i,\ldots,e_f]_{AE}) \geq 2 \wedge$$
$$ef \in Meilleur\_etat_{AE}(e_i) \wedge$$
$$\forall mc_{AE} \in MC_{AE}\ mc_{AE} \not\subset [e_i,\ldots,e_f]_{AE}$$

La figure C.7 présente un exemple de l'ensemble des meilleurs chemins construits pour un arbre d'états. Comme illustré sur cet exemple, deux meilleurs chemins peuvent avoir le même état initial mais ils peuvent aussi avoir des états initiaux différents.

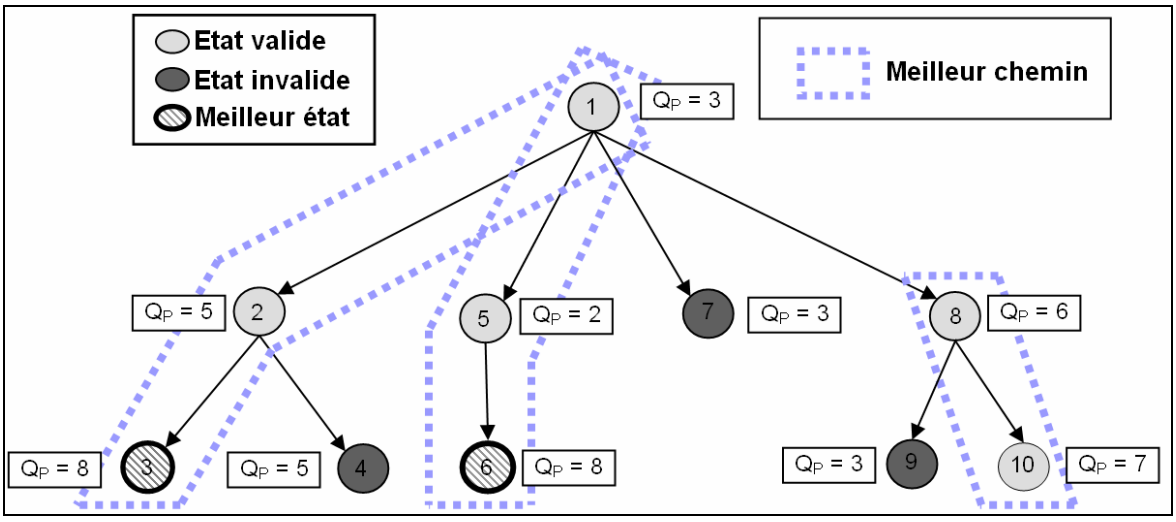


Figure C.7 Exemple d'ensemble des meilleurs chemins pour un arbre d'états

Disposer de l'ensemble des *meilleurs chemins* permet de tirer des informations sur les branches à explorer pour arriver, à partir d'un état donné, au meilleur état de l'arbre (ou du sous-arbre), ainsi que des informations sur l'utilité d'un état. Un état n'appartenant pas à un *meilleur chemin* n'est pas indispensable pour obtenir le meilleur état de l'arbre et est donc dit *inutile*.

Ainsi, à partir de l'état donné d'un meilleur chemin, les décisions qui mènent à l'état suivant du même meilleur chemin (ou d'un autre meilleur chemin partageant le même état initial) sont considérées comme bonnes, alors que les décisions qui mènent à tout autre état sont considérées comme mauvaises. Par exemple, dans la figure C.7, les décisions qui ont permis de passer de l'état 1 à l'état 2 ou à l'état 3 sont bonnes alors que les décisions qui ont entraîné le passage de l'état 1 à l'état 8 sont mauvaises.

C.IV.2.3 Construction de l'ensemble des meilleurs chemins

Nous avons présenté dans la partie précédente (C.IV.2.2) l'intérêt que pouvait avoir la construction de l'ensemble des meilleurs chemins pour la construction des bases d'exemples. Nous présentons dans cette partie (C.IV.2.3), un algorithme que nous proposons pour la construction de l'ensemble des meilleurs chemins d'un arbre d'états.

Le principe de l'algorithme que nous proposons est le suivant : l'ensemble des états est parcouru par ordre décroissant de Q_P. L'état courant est utilisé comme état final d'un nouveau chemin. Le principe de la construction d'un chemin est de partir du dernier état de ce chemin, puis de remonter dans l'arbre, prédécesseur par prédécesseur (en ajoutant chaque état rencontré dans le chemin courant ainsi que dans l'ensemble des états déjà traités) jusqu'à atteindre un état appartenant déjà à l'ensemble des états déjà traités qui n'appartient pas à un meilleur chemin déjà construit dont la qualité du meilleur état est égale à la qualité du meilleur état du chemin courant. Le fait de continuer à remonter dans l'arbre d'états lorsque l'état courant appartient à un meilleur chemin déjà composé d'un meilleur état de même qualité permet de gérer le cas où plusieurs états ont la même valeur de qualité. En effet, comme nous l'avons vu dans la figure C.7, deux meilleurs chemins proposant tous les deux, un état final de même qualité, peuvent avoir des états en commun. Une fois sa construction terminée, le chemin est ajouté à la liste des meilleurs chemins s'il comprend au moins deux états.

```
ConstruireEnsembleMeilleursChemin(Arbre d'états)
   Initialisation Ensemble d'états etatsTraites ← {} //Ensemble des états
   qui ont déjà été analysés
   Initialisation Ensemble de chemins meilleursChemins ← {} //Ensemble des
   meilleurs chemins construits
   Initialisation Liste d'états etatsChemin ← {} //Chemin en cours de
   construction
   Initialisation Ensemble d'états etatsCheminsPrecedents ← {} //Ensemble
   des états appartenant à des meilleurs chemins dont le meilleur état est
   de même qualité que le meilleur état du chemin en cours de construction
   Initialisation Réel Q_P_chemin ← Q_P(Meilleur_etat(arbre état)) //Qualité
   du meilleur état du chemin en cours de construction

   Pour tous les états E de l'arbre classés par ordre décroissant de Q_P
   faire :
       etatsChemin ← {}
       Si Q_P(E) < Q_P_chemin faire :
            Q_P_chemin ← Q_P(E)
            etatsCheminsPrecedents ← {}
       Fin si
       Tant que [E ∉ etatsTraites] OU [E ∈ etatsCheminsPrecedents] faire :
            Ajouter E à etatsTraites
            Ajouter E à etatsChemin
            Ajouter E à etatsCheminsPrecedents
            E ← état précédent dans l'arbre d'états
       Fin tant que
       Si card(etatsChemin) ≥ 2 faire :
          Ajouter etatsChemin à meilleursChemins
       Fin si
   Fin pour
   Retourner meilleursChemins
```

Une fois l'ensemble des meilleurs chemins établi, il nous est possible de l'utiliser afin de construire nos bases d'exemples et de tirer des informations sur les connaissances considérées.

Nous donnons en partie C.IV.3.1, des exemples, pour le modèle AGENT, d'algorithmes de construction de bases d'exemples à partir de cette liste.

C.IV.3 Evaluation des jeux de mesures

C.IV.3.1 Approche proposée d'évaluation des jeux de mesures

Nous avons présenté dans la partie C.IV.2 comment traduire les informations pouvant être tirées de l'analyse locale des arbres d'états sous la forme de bases d'exemples. Avant d'utiliser ces bases d'exemples pour réviser les connaissances, nous proposons de les utiliser pour évaluer la qualité a priori des jeux de mesures.

Nous allons, durant cette phase d'évaluation des jeux de mesures, appliquer des transformations à nos bases d'exemples. Ces transformations sont temporaires et propres à notre phase d'évaluation des jeux de mesures. Les bases d'exemples qui seront utilisées durant le processus de révision des connaissances sont celles présentées en partie C.IV.2 et non celles obtenues à la fin de cette phase d'évaluation.

Nous rappelons que nous nous intéressons dans cette partie aux connaissances qui dépendent de l'état courant de l'entité. Nous avons posé comme formalisation dans la partie C.IV.1, que chaque connaissance est liée à un jeu de mesures permettant de caractériser les états de l'entité. Si le jeu de mesures s'avère peu pertinent, la caractérisation des états sera mauvaise et, quel que soit le processus de révision employé, la connaissance obtenue ne pourra jamais être bonne (cf. B.IV.2). Il est donc important de disposer, pour chaque connaissance à réviser, d'un jeu de mesures pertinentes pour caractériser les états.

Nous proposons d'utiliser les bases d'exemples construites pour évaluer la qualité des jeux de mesures et de fournir une évaluation par connaissance. Notre approche d'évaluation est basée sur la caractérisation des incohérences présentes dans les bases d'exemples. On entend par incohérence, les cas où pour un même état (décrit par le jeu de mesures), deux conclusions différentes sont proposées.

Nous notons K, une connaissance, JM_K, le jeu de mesures lié à cette connaissance et B_K, la base d'exemples construite pour cette connaissance en utilisant le jeu de mesures JM_K pour décrire les états des exemples.

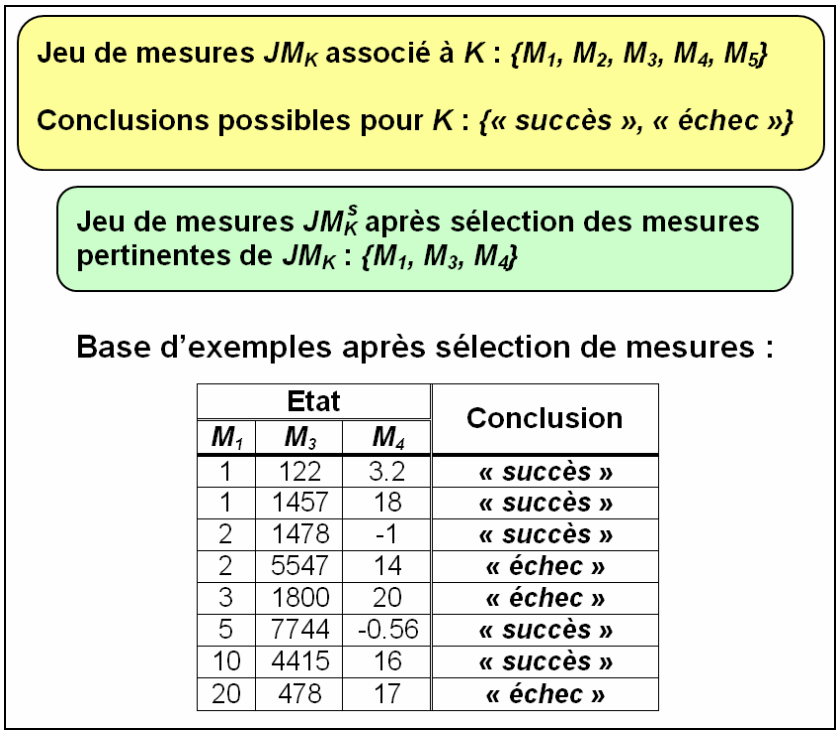
Notre approche d'évaluation d'un jeu de mesures est décomposée en trois étapes :
- Sélection des mesures pertinentes
- Discrétisation des mesures sélectionnées
- Calcul de l'incohérence de la base d'exemples résultante

C.IV.3.2 Sélection des mesures pertinentes

Les exemples de nos bases d'exemples sont représentés dans un formalisme attributs/valeur très utilisé en apprentissage supervisé. Il est donc possible de recourir aux nombreux travaux qui ont porté sur la sélection d'attributs dans le cadre de l'apprentissage supervisé (Caruana & Freitag, 1994 ; Hall & Smith, 1998 ; Guyon & Elisseeff, 2003 ; Hall & Holmes, 2003). Il existe de nombreux algorithmes permettant de sélectionner, parmi un ensemble d'attributs, les plus intéressants pour l'apprentissage. L'intérêt d'une telle sélection est de faciliter l'apprentissage et d'améliorer l'interprétabilité des modèles prédictifs générés. Nous présentons en annexe *An.2.4* un exemple de d'algorithme de sélection d'attributs.

Nous proposons d'utiliser ces techniques pour notre problème d'évaluation des jeux de mesures en vue d'éliminer les mesures peu pertinentes qui n'apportent pas d'informations utiles sur l'état de l'entité.

L'application d'un algorithme de ce type sur la base d'exemples liée à la connaissance K permet d'obtenir un jeu de mesures $JM_K^s \subset JM_K$. La figure C.8 donne un exemple de sélection de mesures pertinentes par rapport à la base d'exemples présentée figure C.6.



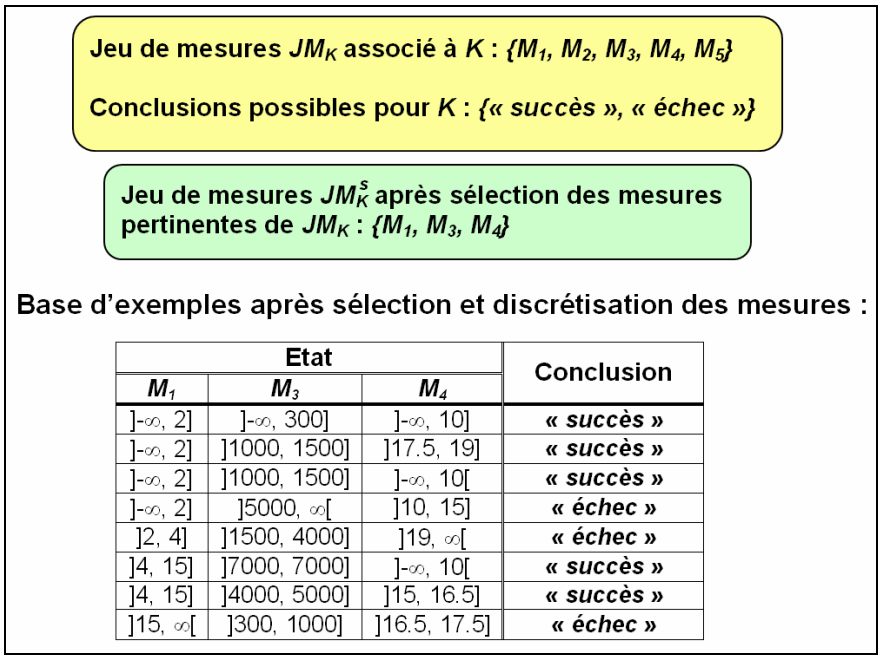
Etat			Conclusion
M_1	M_3	M_4	
1	122	3.2	« succès »
1	1457	18	« succès »
2	1478	-1	« succès »
2	5547	14	« échec »
3	1800	20	« échec »
5	7744	-0.56	« succès »
10	4415	16	« succès »
20	478	17	« échec »

Figure C.8 Exemple de sélection de mesures pertinentes

C.IV.3.3 Discrétisation des mesures pertinentes

De même que pour la sélection d'attributs, de nombreux travaux ont porté sur la discrétisation d'attributs numériques dans le cadre de l'apprentissage supervisé (Fayyad & Irani, 1992 ; Dougherty et al., 1995 ; Boullé, 2006). De nombreux algorithmes permettent, par analyse d'un jeu d'apprentissage, de passer d'un attribut continu à un attribut catégoriel constitué d'un ensemble d'intervalles disjoints. Nous présentons en annexe *An.2.5* un exemple d'algorithme de discrétisation d'attributs.

L'enjeu, pour notre approche, d'utiliser de telles techniques est de diminuer la taille de l'espace des jeux de mesures et de rendre fini le nombre d'états différents possibles pour le jeu de mesures. En effet, certaines mesures peuvent admettre des variations faibles de valeurs sans que ces modifications justifient un traitement différent de l'entité. Par exemple si l'on prend l'exemple de deux bâtiments dont la seule différence est que l'un a une aire de 198.58 m^2 et l'autre de 198.6 m^2, il y a de fortes chances pour que ces deux bâtiments soient généralisés exactement de la même manière et il n'est pas intéressant de faire une distinction entre ces deux valeurs d'aire. Evidemment, la discrétisation des mesures ne concerne que les mesures à valeur quantitative non celles à valeur qualitative.
La figure C.9 donne un exemple de discrétisation d'un jeu de mesures pour la base d'exemples présentée figure C.8.



Etat			Conclusion
M_1	M_3	M_4	
]-∞, 2]	]-∞, 300]	]-∞, 10]	« succès »
]-∞, 2]	]1000, 1500]	]17.5, 19]	« succès »
]-∞, 2]	]1000, 1500]	]-∞, 10[	« succès »
]-∞, 2]	]5000, ∞[	]10, 15]	« échec »
]2, 4]	]1500, 4000]	]19, ∞[	« échec »
]4, 15]	]7000, 7000]	]-∞, 10[	« succès »
]4, 15]	]4000, 5000]	]15, 16.5]	« succès »
]15, ∞[	]300, 1000]	]16.5, 17.5]	« échec »

Figure C.9 Exemple de discrétisation de jeu de mesures

C.IV.3.4 Calcul de l'incohérence de la base d'exemples résultantes

Nous obtenons, après discrétisation des mesures pertinentes, une nouvelle base d'exemples que nous utilisons pour évaluer le jeu de mesures JM_K. Nous proposons pour cela de calculer le taux d'incohérence de la base d'exemples obtenue. Une base d'exemples totalement incohérente est une base dans laquelle, pour chaque exemple, chaque étiquette a la même valeur alors que les mesures ont des valeurs différentes.

Nous définissons l'ensemble d'états E_{B_K} constitué de l'ensemble des états différents, en termes de valeurs de mesures, contenus dans la base d'exemples B_K. E_{B_K} contient donc un exemplaire unique, en terme de valeurs de mesures, de chaque état contenu dans la base d'exemples B_K.

Nous définissons également la fonction $nb\_exemples_{B_K}(e,c)$ qui renvoie le nombre d'exemples de la base d'exemples B_K qui ont pour état l'état e et pour conclusion, la conclusion c.

Le taux de cohérence d'une base d'exemples B_K correspond à la proportion d'exemples présentant majoritairement une conclusion c pour chaque état e, par rapport au nombre total d'exemples de notre base. Nous proposons donc comme formule pour le taux d'incohérence d'une base d'exemples B_K liée à une connaissance K dont l'ensemble des conclusions possibles est C_K :

$$Incohérence(B_K) = 1 - \frac{\sum_{e \in E_{B_K}} \left[\max_{c \in C_K} (nb\_exemples_{B_K}(e,c)) \right]}{card(E_{B_K})}$$

Nous pouvons borner le taux d'incohérence pour une base d'exemples B_K :

$$0 \leq Incohérence(B_K) \leq 1 - \frac{1}{card(C_K)}$$

En effet, au mieux, pour chaque état différent de la base d'exemples, une seule conclusion est proposée est on a :

$$\sum_{e \in E_{B_K}} \left[\max_{c \in C_K} (nb\_exemples_{B_K}(e,c)) \right] = card(E_{B_K})$$

Le taux d'incohérence est alors égal à 0 ce qui signifie que la base d'exemples est totalement cohérente.

Dans le pire des cas, pour chaque état différent de la base d'exemples, toutes les conclusions possibles sont représentées autant de fois dans B_K. Nous pouvons alors poser que pour chaque état *e* différent :

$$\max_{c \in C_K} (nb\_exemples_{B_K}(e,c)) = \frac{1}{card(C_K)}$$

Le taux d'incohérence est donc égal à $1 - \frac{1}{card(C_K)}$, ce qui signifie que la base est totalement incohérente.

Plus une base d'exemples a un taux d'incohérence élevé, plus celle-ci souffre d'un bruit de description et donc plus il est important d'ajouter de nouvelles mesures pertinentes pour caractériser les états.

C.IV.4 Application pour le système AGENT

C.IV.4.1 Construction des bases d'exemples

C.IV.4.1.1 Introduction

Nous avons présenté en partie C.IV.2.2 notre approche générale de construction de bases d'exemples. Notre approche se base sur l'analyse des arbres d'états construits durant la phase d'exploration. Nous avons évoqué le fait que la construction des bases d'exemples est dépendante du type de connaissances considéré. Ainsi, il n'est pas possible de fournir d'algorithme générique capable de construire une base d'exemples pour tout type de connaissances. En vue de trouver des points communs entre les différents processus de construction de bases d'exemples, nous avons proposé de baser la construction des bases d'exemples sur l'ensemble des meilleurs chemins. Cet ensemble, qui est établi pour chaque arbre d'états, donne des indications sur les décisions à prendre à partir d'un état de l'arbre d'états. Nous avons proposé en partie C.IV.2.3 un algorithme de construction de cet ensemble des meilleurs chemins pour un arbre d'états donné.

Nous présentons dans cette partie C.IV.4, les algorithmes que nous proposons pour la construction des différentes bases d'exemples utilisées dans le cadre de la révision des connaissances du modèle AGENT. Ce modèle comprend six types de connaissances

différents : les connaissances sur les priorités, sur l'application des actions, sur la restriction des actions, sur la fin de cycle, sur la validité des états et sur l'optimalité des états (cf. B.V.3.1.2). Parmi ces six types de connaissances, nous n'utiliserons des bases d'exemples que pour quatre d'entre eux. En effet, les connaissances de validité des états et celles de restrictions d'application des actions ne dépendent pas d'une situation locale (l'état courant de l'agent géographique) mais de la situation globale (la façon dont l'arbre d'états est parcouru) (cf. C.IV.1).

Nous notons *Val(E, JM_K),* le vecteur de valeurs obtenues pour les mesures du jeu de mesures JM_K pour un état *E* d'un arbre d'états. Nous notons de même, *Init(etatsChemin)* l'état initial du chemin *etatsChemin*.

C.IV.4.1.2 Connaissances relatives à la priorité des contraintes

En ce qui concerne les connaissances relatives à la priorité des contraintes, une base d'exemples est construite par contrainte. Une base d'exemples supplémentaire commune à l'ensemble des connaissances sur la priorité des contraintes est construite.

Nous rappelons que la priorité d'une contrainte traduit, pour un état donné de l'agent géographique, le niveau d'urgence de traitement de la contrainte. Plus une contrainte a une priorité élevée pour un état, plus les actions qu'elle propose seront appliquées en priorité par l'agent géographique.

Les exemples des bases d'exemples sont composés d'un état et d'une conclusion. Dans le cas des bases d'exemples liées à la connaissance sur la priorité d'une contrainte, l'état d'un exemple correspond à un état d'un arbre d'états caractérisé par la satisfaction de la contrainte considérée. Pour les conclusions des exemples, deux conclusions sont possibles : soit « prioritaire » qui indique que la contrainte doit être prioritaire pour l'état considéré, soit « non prioritaire » qui indique que la contrainte ne doit pas l'être.
Concernant la base d'exemples commune à l'ensemble des connaissances sur la priorité des contraintes, l'état d'un exemple correspond à un état d'un arbre d'états caractérisé par les satisfactions de l'ensemble des contraintes. La conclusion affectée à un état correspond à l'ensemble des contraintes devant être prioritaires pour cet état.

Le fait qu'une contrainte doive ou non être prioritaire pour un état donné dépend des états obtenus après application des actions proposées par la contrainte. Ainsi, si pour un état donné appartenant à un ou plusieurs meilleurs chemins l'application d'au moins une des actions proposées par la contrainte a conduit à un état appartenant à l'un de ces chemins, la contrainte est considérée comme devant être prioritaire. Dans le cas contraire, la contrainte est considérée comme ne devant pas être prioritaire.

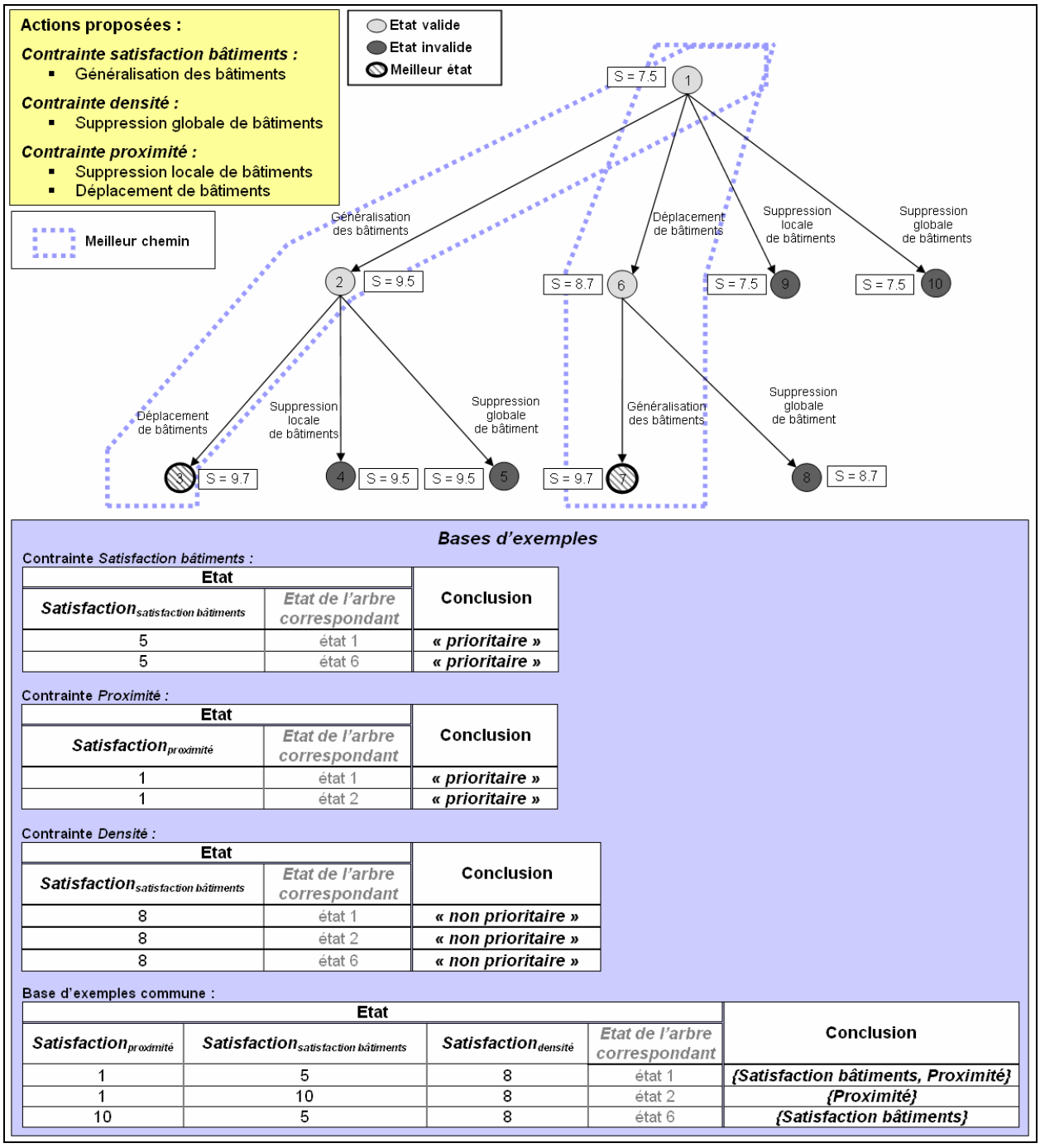
Afin d'optimiser le temps nécessaire à la construction des bases d'exemples liées à la priorité des contraintes, nous proposons de construire l'ensemble des bases d'exemples en même temps.

Nous notons $E_{succ}(E, Ctr)$, l'ensemble des états successeurs de *E* par application d'une action proposée par la contrainte *Ctr*. Nous notons $Satisfaction_{Ctr}$ pour désigner la satisfaction de la contrainte *Ctr*, et $\bigcup_{ctr} satisfaction_{Ctr}$ le jeu de mesures réunissant l'ensemble des satisfactions des contraintes.

L'algorithme que nous proposons pour la construction des bases d'exemples est basé sur le principe suivant : chaque état appartenant à un meilleur chemin est analysé. Si, à partir d'un de ces états, au moins l'une des actions proposées par une contrainte mène à un autre état du même meilleur chemin ou à d'un autre meilleur chemin ayant le même état initial, l'utilisation de cette contrainte comme contrainte prioritaire est considérée comme un succès. Si au contraire toutes les actions proposées par la contrainte mènent à un état inutile (état n'appartenant pas au même meilleur chemin ou à un autre meilleur chemin ayant le même état initial), l'utilisation de cette contrainte comme contrainte prioritaire est considérée comme un échec.

```
ConstruireBasesExemplesPriorite(Arbre d'états)
  Pour tous les états E de l'arbre faire :
    Si E ∈ etatsChemin avec etatsChemin ∈ meilleursChemins faire :
       Initialisation ctrsPrioriraitaires ← {}
       Pour toutes les contraintes ctr de l'agent géographique faire :
          Si E_succ(E, ctr) ≠ {} faire :
             Pour tous E_succ ∈ E_succ(E, ctr) faire :
                Si [E_succ ∈ etatsChemin] OU [E_succ ∈ etatsChemin' avec
                Init(etatsChemin') = Init(etatsChemin)] faire :
                   Ajouter ctr à ctrsPrioriraitaires
                Fin si
             Fin pour
          Fin si
       Fin pour
       Ajouter, dans la base d'exemples commune, un exemple ayant pour
       état Val(E, ⋃_ctr satisfaction_Ctr ) et pour conclusion ctrsPrioriraitaires
       Pour toutes les contraintes ctr de l'agent géographique faire :
          Si (ctr est inclus dans ctrsPrioriraitaires)
             Ajouter, dans la base d'exemples de ctr, un exemple ayant
             pour état Val(E, Satisfaction_Ctr) et pour conclusion
             « prioritaire »
          Sinon faire :
             Ajouter, dans la base d'exemples de ctr, un exemple ayant
             pour état Val(E, Satisfaction_Ctr) et pour conclusion « non
             prioritaire »
          Fin si
       Fin pour
    Fin si
  Fin pour
```

La figure C.10 donne un exemple de bases d'exemples obtenues pour les connaissances de priorité des contraintes après analyse d'un arbre d'état. Pour des raisons de clarté de la figure, nous n'avons pas mis à côté de chaque état la valeur des satisfactions de ses contraintes, mais nous avons par contre ajouté dans les bases d'exemples, la colonne « *Etat de l'arbre correspondant* » qui donne l'état de l'arbre auquel se réfère chaque exemple.



Bases d'exemples

Contrainte *Satisfaction bâtiments* :

Etat		Conclusion
$Satisfaction_{satisfaction\ bâtiments}$	*Etat de l'arbre correspondant*	
5	état 1	*« prioritaire »*
5	état 6	*« prioritaire »*

Contrainte *Proximité* :

Etat		Conclusion
$Satisfaction_{proximité}$	*Etat de l'arbre correspondant*	
1	état 1	*« prioritaire »*
1	état 2	*« prioritaire »*

Contrainte *Densité* :

Etat		Conclusion
$Satisfaction_{satisfaction\ bâtiments}$	*Etat de l'arbre correspondant*	
8	état 1	*« non prioritaire »*
8	état 2	*« non prioritaire »*
8	état 6	*« non prioritaire »*

Base d'exemples commune :

Etat				Conclusion
$Satisfaction_{proximité}$	$Satisfaction_{satisfaction\ bâtiments}$	$Satisfaction_{densité}$	*Etat de l'arbre correspondant*	
1	5	8	état 1	*{Satisfaction bâtiments, Proximité}*
1	10	8	état 2	*{Proximité}*
10	5	8	état 6	*{Satisfaction bâtiments}*

Figure C.10 Exemple de bases d'exemples construites pour les connaissances sur la priorité des contraintes

C.IV.4.1.3 Connaissances relatives à l'application des actions

En ce qui concerne les connaissances relatives à l'application des actions, une base d'exemples est construite par action. De la même manière que pour les connaissances sur la priorité des contraintes, nous construisons toutes les bases d'exemples en même temps.
Nous rappelons que les connaissances d'application des actions permettent de définir, pour un état donné de l'agent géographique, quelles actions doivent être appliquées et avec quel poids.

Plus une action à un poids élevé pour un état, plus l'action sera appliquée en priorité par rapport aux autres actions proposées par la même contrainte.

Dans le cas des bases d'exemples liées à la connaissance sur l'application d'une action, l'état d'un exemple correspond à un état d'un arbre d'états caractérisé par le jeu de mesures lié à la contrainte qui a proposé l'action. Ainsi, si une contrainte propose plusieurs actions les exemples construits pour les connaissances d'application de ces actions seront caractérisés par un même jeu de mesures. Pour les conclusions des exemples, deux conclusions sont possibles : soit « *appliquer* », qui indique que l'action doit bien être appliquée pour l'état considéré, soit « *ne pas appliquer* », qui indique que l'action ne doit pas l'être.
Concernant la base d'exemples commune à l'ensemble des connaissances sur l'application des actions, l'état d'un exemple correspond à un état d'un arbre d'états caractérisé par l'ensemble des mesures liées à chacune des contraintes proposant des actions. La conclusion affectée à un état correspond à l'ensemble des actions devant être appliquées pour cet état.

Le fait qu'une action doive ou non être appliquée pour un état donné dépend de l'état obtenu après son application. Ainsi, si pour un état donné, appartenant à un ou plusieurs meilleurs chemins, l'application de l'action a conduit à un état appartenant à l'un de ces chemins, l'action est considérée comme devant être appliquée. Dans le cas contraire, l'action est considérée comme ne devant pas être appliquée.

Nous notons $Act(E, E_{succ})$ l'action ayant permis par son application de passer de l'état E à l'état E_{succ}. Nous notons de même JM_{KAct} le jeu de mesures lié à une action Act (c'est-à-dire lié à la contrainte qui propose l'action Act), et $\bigcup_{Act} JM_{KAct}$, le jeu de mesures réunissant l'ensemble des jeux de mesures liés aux actions.

Le principe de l'algorithme que nous proposons pour la construction des bases d'exemples est le suivant : chaque état appartenant à un meilleur chemin est analysé. Si, à partir d'un de ces états, l'action proposée mène à un autre état du même meilleur chemin ou d'un autre meilleur chemin ayant le même état initial, l'application de cette action est considérée comme un succès. Si l'action proposée mène à un état inutile (état n'appartenant pas au même meilleur chemin ou à un autre meilleur chemin ayant le même état initial), l'action est considérée comme un échec.

```
ConstruireBasesExemplesActions(Arbre d'états)
  Pour tous les états E de l'arbre faire :
    Si E ∈ etatsChemin avec etatsChemin ∈ meilleursChemins faire :
       Initialisation actsAppliquées ← {}
       Pour tous les états successeurs E_succ dans l'arbre faire :
          Initialisation Action A ← Act(E,E_succ)
          Si [E_succ ∈ etatsChemin] OU [E_succ ∈ etatsChemin' avec état initial
          de (etatsChemin') = état initial de (etatsChemin)] faire :
               - Ajouter l'action A à actsAppliquées
               - Ajouter, dans la base d'exemples de l'action A, un exemple
               ayant pour état Val(E,JM_KA) et pour conclusion « appliquer »
          Sinon faire :
              Ajouter, dans la base d'exemples de l'action A, un exemple
              ayant pour état Val(E,JM_KA) et pour conclusion « ne pas
              appliquer »
          Fin si
       Fin pour
       Ajouter, dans la base d'exemples commune, un exemple ayant pour état
       Val(E, ⋃_Act JM_KAct ) et pour conclusion actsAppliquées
    Fin si
  Fin pour
```

La figure C.11 donne un exemple de bases d'exemples obtenues pour les connaissances d'application des actions après analyse d'un arbre d'état.
A noter que nous utilisons dans cet exemple ainsi que dans les exemples suivants des notations *génériques* pour les mesures (M_1, M_2, M_3, M_4) afin de mettre en valeur le fait que les algorithmes que nous proposons dans cette partie C.IV.4.1 sont génériques et peuvent être utilisés pour n'importe quelle classe d'agents géographiques du modèle AGENT. Le choix des jeux de mesures utilisés pour caractériser les états dépendra des types d'agents géographiques et de contraintes considérés.

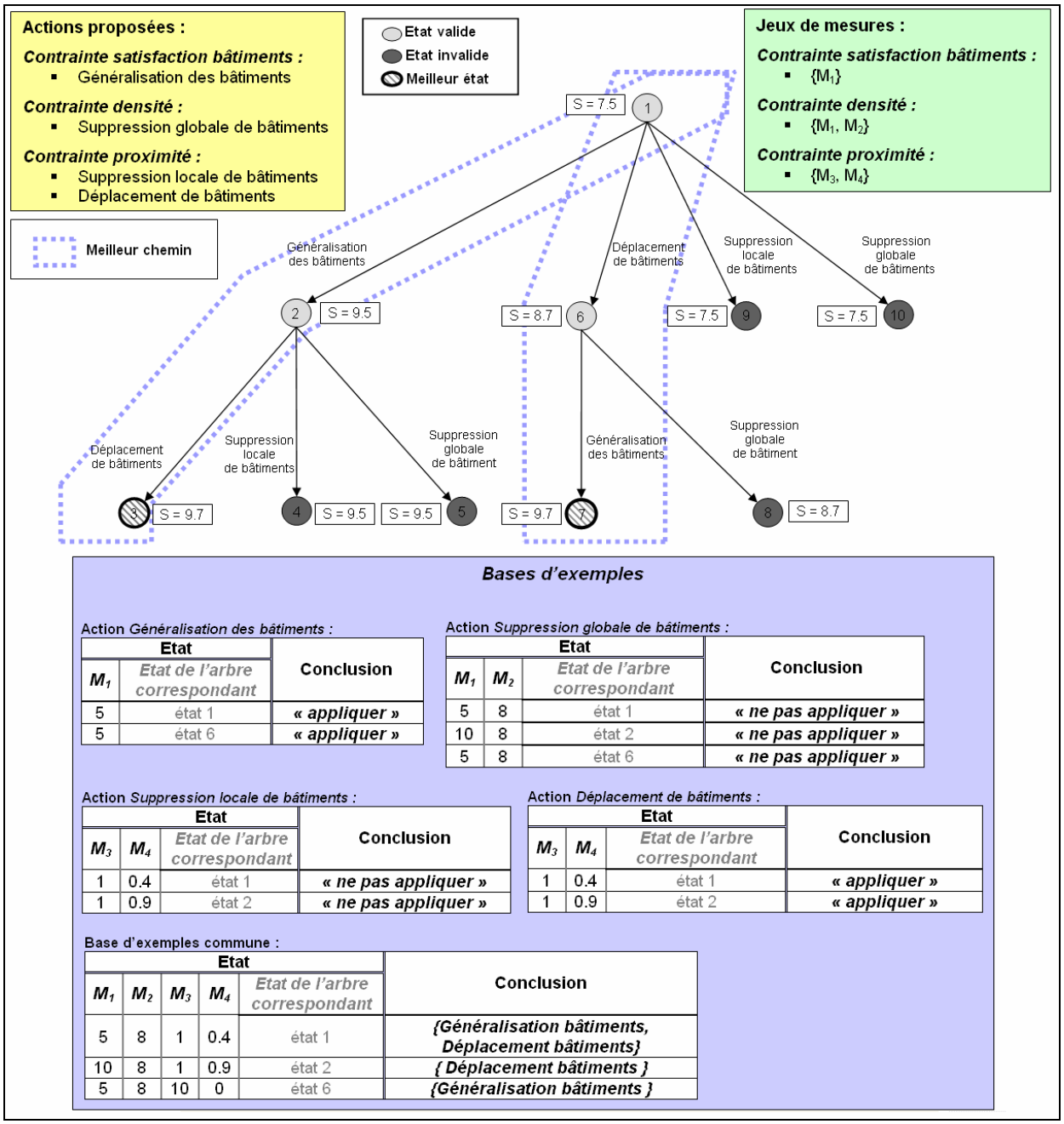


Figure C.11 Exemple de bases d'exemples construites pour les connaissances sur l'application des actions

C.IV.4.1.4 Connaissance relative à l'optimalité des états

Nous rappelons qu'un état optimal est un état *e* dont la qualité (la satisfaction) est supérieure ou égale à celles de tous les états appartenant au sous-arbre dont la racine est l'état *e* (cf. B.V.3.1.1)

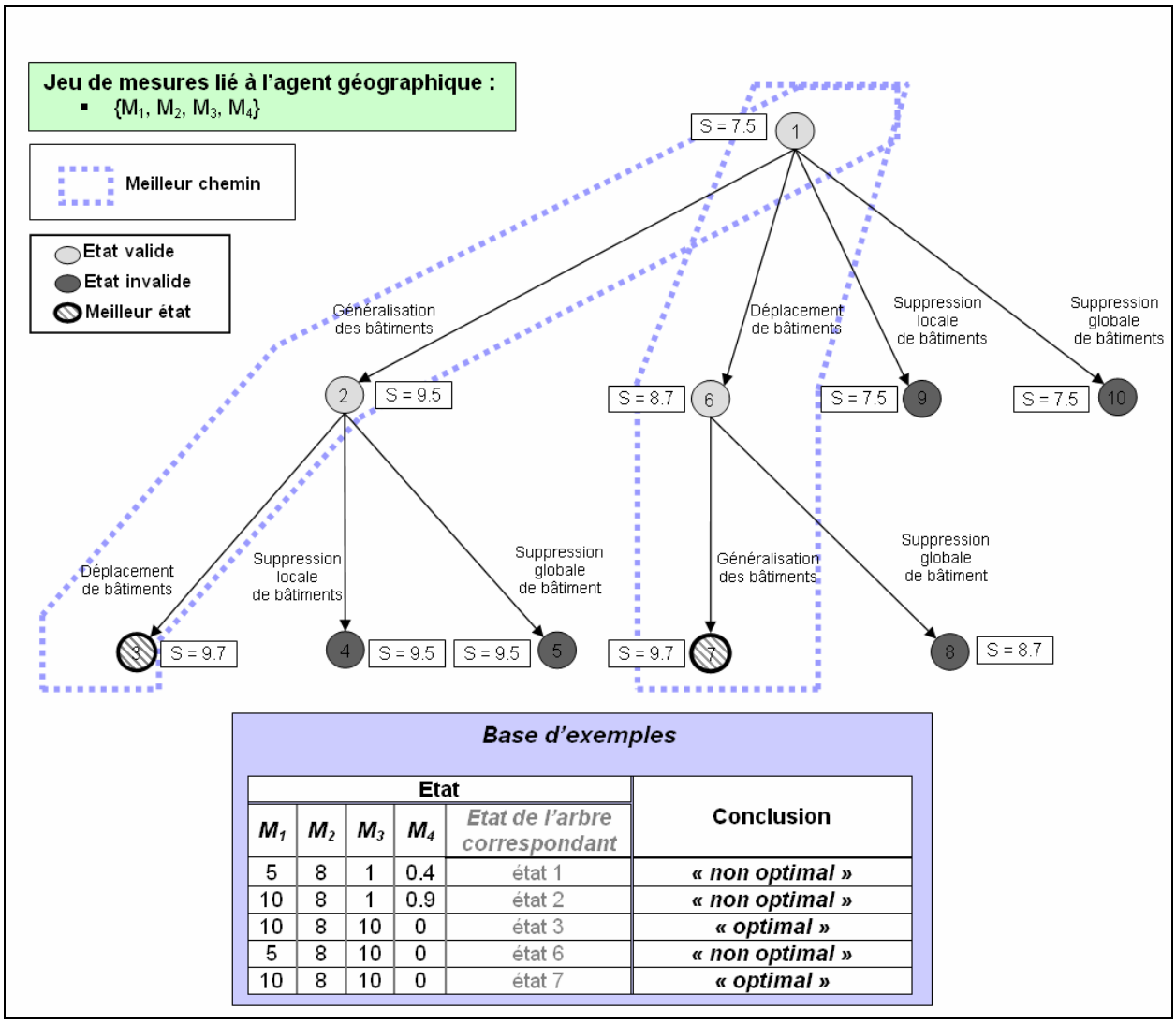
L'état d'un exemple pour cette connaissance correspond à un état d'un arbre d'états caractérisé par le jeu de mesures lié à l'agent géographique. La conclusion d'un exemple peut, elle, prendre deux valeurs différentes : « optimal » lorsque l'état est optimal et « non optimal » lorsqu'il ne l'est pas.

Nous notons de même JM_{Ag} le jeu de mesures lié à un agent géographique *Ag*. Nous notons $Satisfaction_{Ag}(e)$, la fonction renvoyant la satisfaction de l'agent *Ag* pour l'état *e*.

Le principe de l'algorithme que nous proposons pour la construction de la base d'exemples est le suivant : les états sont classés par ordre décroissant de satisfaction. Cette liste d'états est parcourue en partant du meilleur pour aller au moins bon. Nous testons pour l'état courant si aucun des états qui ont déjà été définis comme *optimaux* n'est descendant de cet état courant et n'a de satisfaction supérieure. S'il n'a pas de descendant meilleur que lui, alors l'état courant est *optimal*.

```
ConstruireBaseExemplesOptimalite(Arbre d'états)
  Initialisation Ensemble etatsOptimaux ← {}
  Pour tous les états E de l'arbre classés par ordre décroissant de
  satisfaction faire :
     Initialisation Booléen estOptimal ← VRAI
     Pour tous les états E_opt appartenant à etatsOptimaux faire :
        Si [E_opt est un descendant de E] ET
           [Satisfaction_Ag(E_opt) > Satisfaction_Ag(E)]faire :
              estOptimal ← FAUX
        Fin si
     Fin pour
     Si (estOptimal = VRAI) faire :
        - Ajouter E à etatsOptimaux
        - Ajouter, dans la base d'exemples, un exemple ayant pour état
        Val(E,JM_K) et pour conclusion « optimal »
     Sinon faire :
        - Ajouter, dans la base d'exemples, un exemple ayant pour état
        Val(E,JM_K) et pour conclusion « non optimal »
     Fin si
  Fin pour
```

La figure C.12 propose un exemple de base d'exemples obtenue. Nous rappelons que dans le cadre de la connaissance d'optimalité, le jeu de mesures utilisé est celui qui est lié à l'agent géographique.



Base d'exemples

Etat					Conclusion
M_1	M_2	M_3	M_4	*Etat de l'arbre correspondant*	
5	8	1	0.4	état 1	***« non optimal »***
10	8	1	0.9	état 2	***« non optimal »***
10	8	10	0	état 3	***« optimal »***
5	8	10	0	état 6	***« non optimal »***
10	8	10	0	état 7	***« optimal »***

Figure C.12 Exemple de base d'exemples construite pour la connaissance sur l'optimalité des états

C.IV.4.1.5 Connaissance relative à la fin de cycle

Nous nous intéressons ici à la construction de la base d'exemples liée à la connaissance sur la fin de cycle. La base d'exemples pour cette connaissance est construite par analyse de l'ensemble des meilleurs chemins.

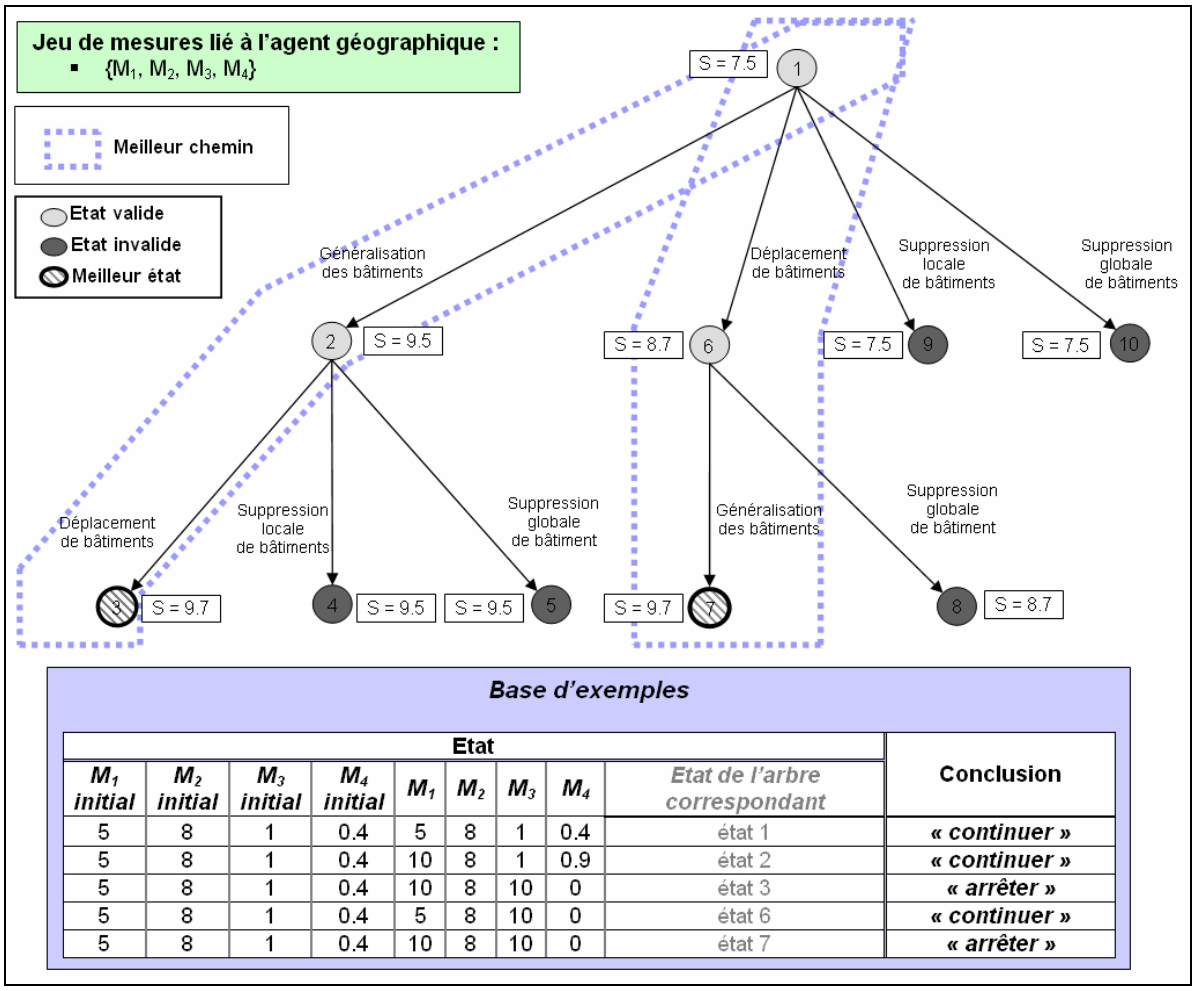
L'état d'un exemple correspond à deux états de l'arbre d'états caractérisés par le jeu de mesures lié à l'agent géographique. En effet, la connaissance sur la fin de cycle dépend à la fois de l'état courant mais également de l'état initial de l'agent géographique. L'état d'un exemple correspond donc à un vecteur de valeurs regroupant les valeurs prises par les mesures du jeu de mesures lié à l'agent géographique pour un état d'un meilleur chemin ainsi que pour l'état initial de ce meilleur chemin.

La conclusion d'un exemple peut, quant à elle, prendre deux valeurs différentes : « continuer » lorsque l'état courant n'est pas le meilleur état du meilleur chemin considéré et « arrêter » lorsque l'état courant est le meilleur état du meilleur chemin considéré.

Nous notons $Val(E_1, E_2, JM_K)$, le vecteur de valeurs composé des valeurs obtenues pour les mesures du jeu de mesures JM_K pour les états E_1 et E_2 d'un arbre d'états.
Le principe de l'algorithme que nous proposons est le suivant : chaque meilleur chemin est parcouru. Tant que le meilleur état de ce chemin n'est pas atteint, le cycle d'actions est considéré comme devant se poursuivre. Une fois le meilleur état atteint, il est considéré comme devant s'arrêter.

```
ConstruireBasesExemplesFinCycle(Arbre d'états)
  Pour tous les états E de l'arbre classés par ordre décroissant de
  satisfaction faire :
     Pour tous les meilleurs chemins etatsChemin ∈ meilleursChemins faire :
        Si E ∈ etatsChemin faire :
           Si E est le meilleur état de etatsChemin faire :
              Ajouter, dans la base d'exemples, un exemple ayant pour état
              Val(Init(etatsChemin),E,JM_K) et pour conclusion « arrêter »
           Sinon faire :
              Ajouter, dans la base d'exemples, un exemple ayant pour état
              Val(Init(etatsChemin),E,JM_K) et pour conclusion « continuer »
           Fin si
        Fin si
     Fin pour
  Fin pour
```

La figure C.13 présente un exemple de base d'exemples obtenue. Nous rappelons que dans le cadre de la connaissance de fin de cycle, le jeu de mesures utilisé est celui qui est lié à l'agent géographique.



Base d'exemples

Etat									Conclusion
M_1 initial	***M_2 initial***	***M_3 initial***	***M_4 initial***	***M_1***	***M_2***	***M_3***	***M_4***	*Etat de l'arbre correspondant*	
5	8	1	0.4	5	8	1	0.4	état 1	***« continuer »***
5	8	1	0.4	10	8	1	0.9	état 2	***« continuer »***
5	8	1	0.4	10	8	10	0	état 3	***« arrêter »***
5	8	1	0.4	5	8	10	0	état 6	***« continuer »***
5	8	1	0.4	10	8	10	0	état 7	***« arrêter »***

Figure C.13 Exemple de base d'exemples construite pour la connaissance sur la fin de cycle

C.IV.4.2 Evaluation des jeux de mesures

Nous avons présenté en partie C.IV.3 une approche pour évaluer la qualité d'un jeu de mesures. Cette approche, basée sur le calcul du taux d'incohérence de la base d'exemples, peut être appliquée à certaines connaissances du modèle AGENT.

Comme nous l'avons évoqué en partie C.IV.3, cette approche est destinée aux connaissances qui dépendent uniquement d'une situation locale et pour lesquelles nous avons construit des bases d'exemples.

Dans le cadre du modèle AGENT, quatre types de connaissances répondent à cette contrainte : les connaissances sur la priorité des contraintes, les connaissances sur l'application des actions, les connaissances sur l'optimalité des états et les connaissances sur la fin de cycle. Parmi ces quatre types de connaissances, seuls trois dépendent de jeux de mesures qui peuvent être enrichis. En effet, les connaissances relatives à la priorité des contraintes dépendent uniquement de la satisfaction de la contrainte considérée, il n'est pas possible d'ajouter des mesures pour enrichir le domaine de définition de ces connaissances.

Nous appliquerons donc notre approche d'évaluation uniquement aux jeux de mesures utilisés par les connaissances d'application des actions, de fin de cycle et d'optimalité.

Nous proposons dans ce cadre une expérimentation visant à valider notre approche d'évaluation au chapitre F.

C.V Bilan

Notre approche générale de révision des connaissances se base sur l'utilisation de l'expérience passée. Nous avons introduit dans ce chapitre, notre approche de production de l'expérience (figure C.14).

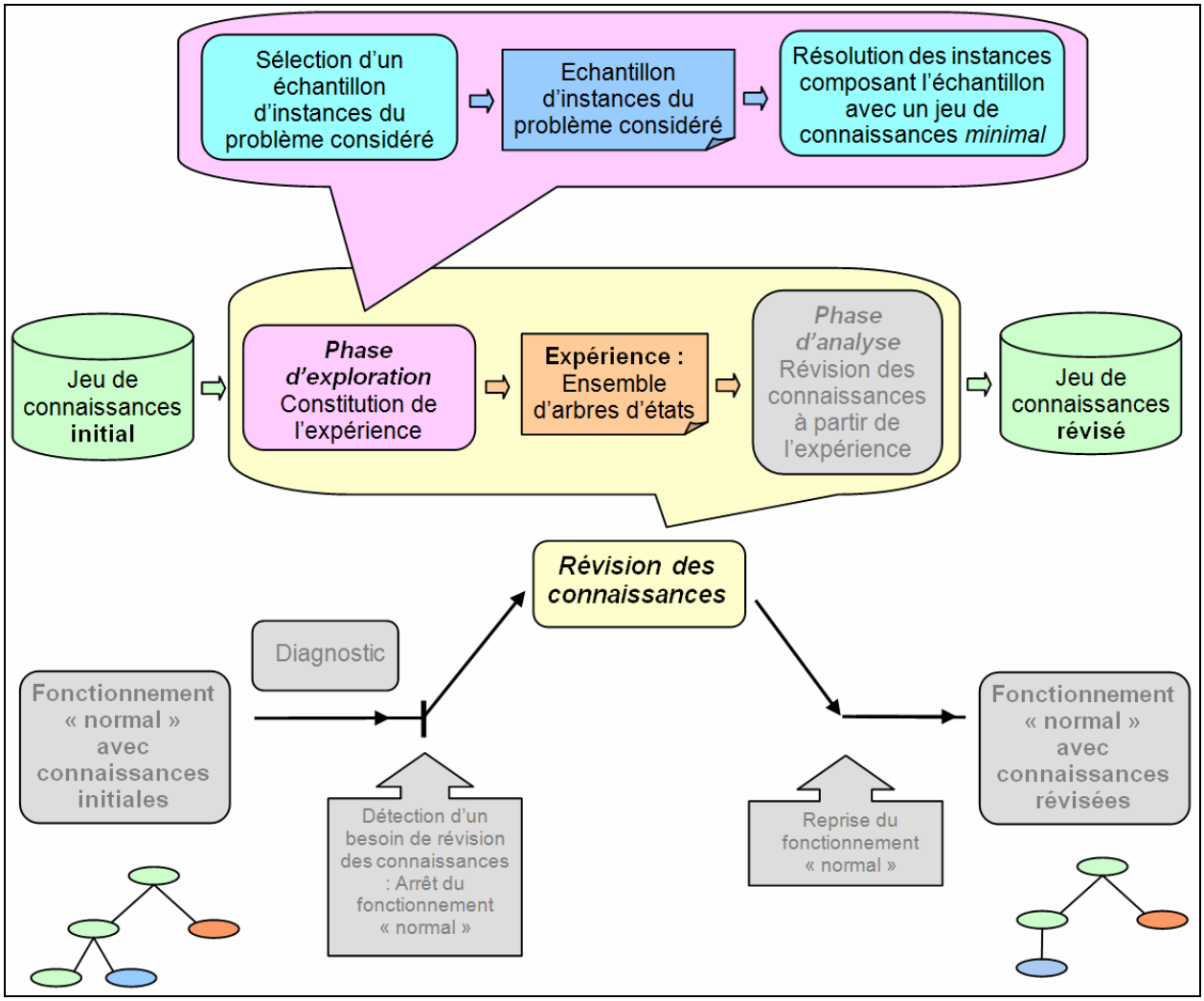


Figure C.14 Approche détaillée de constitution de l'expérience

Notre approche consiste, dans un premier temps, à sélectionner automatiquement un échantillon d'instances du problème considéré. Nous avons ainsi proposé une méthode de sélection automatique d'instances caractéristiques basée sur le partitionnement de l'ensemble des instances disponibles en familles et sur le choix, dans chacune de ces familles, des instances les plus représentatives. Une expérimentation permettant d'évaluer cette méthode sera présentée au chapitre F.

La deuxième étape de notre approche, une fois l'échantillon de révision établi, consiste à résoudre l'ensemble des instances de l'échantillon avec un jeu de connaissances spécifique, dit *minimal*. Nous nous sommes intéressé dans ce chapitre à la définition d'un tel de jeu de

connaissances qui permet de construire l'ensemble des états atteignables par l'ensemble des jeux de connaissances possibles. Une fois une instance de problème résolue avec un tel jeu de connaissances, il devient possible de simuler la résolution de cette instance avec n'importe quel jeu de connaissances sans avoir à refaire de calculs lourds, ce qui se révélera très utile pour notre processus de révision. Nous avons illustré, en partie C.III.2, ce concept de jeu de connaissances *minimal* au travers de la définition d'un tel jeu pour le modèle de généralisation AGENT.

Il est possible, pour certaines connaissances, de recueillir, à partir de l'analyse locale des arbres d'états, des informations supplémentaires relatives aux succès et aux échecs rencontrés par chaque connaissance. Nous nous sommes intéressé en partie C.IV à la collecte de ces informations sous la forme de bases d'exemples. Nous avons également présenté dans cette partie une approche visant à évaluer la qualité des jeux de mesures utilisés pour définir les connaissances par analyse des bases d'exemples.

Nous disposons maintenant d'une méthode automatique de construction de l'expérience. Nous nous intéressons, dans la partie suivante, au processus de révision des connaissances qui se base sur l'analyse de cette expérience.

Chapitre D
Révision des connaissances procédurales par analyse des traces d'exécution

D.I Introduction

Nous avons introduit dans le chapitre précédent, les problématiques liées à la production de l'expérience. Nous avons ainsi présenté une approche visant à sélectionner, parmi un ensemble d'instances disponibles du problème d'optimisation considéré, un échantillon d'instances représentatives. Nous avons également défini les caractéristiques des connaissances à utiliser pour résoudre ces instances. L'expérience acquise durant cette *phase d'exploration*, prend la forme d'un ensemble d'arbres d'états où chaque arbre représente l'ensemble des états pouvant être construits lors de la résolution de l'une des instances de notre échantillon représentatif.

Nous nous intéressons, dans ce chapitre, à la seconde phase de notre processus de révision : la *phase d'analyse* (figure D.1). Celle-ci consiste à utiliser l'expérience acquise en vue de réviser les connaissances procédurales du système de résolution de problèmes. Nous rappelons que les connaissances procédurales désignent les connaissances ayant trait à la façon dont les arbres d'états sont parcourus. Il s'agira typiquement des connaissances permettant de définir quelles actions appliquer pour un état donné et dans quel ordre, si l'état courant est valide ou optimal et si le cycle d'actions doit s'arrêter ou non après avoir visité un état.
Nous désignons le processus mené durant cette *phase d'analyse* par le terme « *processus de révision par analyse* ». Notons qu'il est important de faire une différentiation entre ce processus et le processus global de révision qui comprend non seulement le *processus de révision par analyse*, mais aussi la *phase d'exploration*.

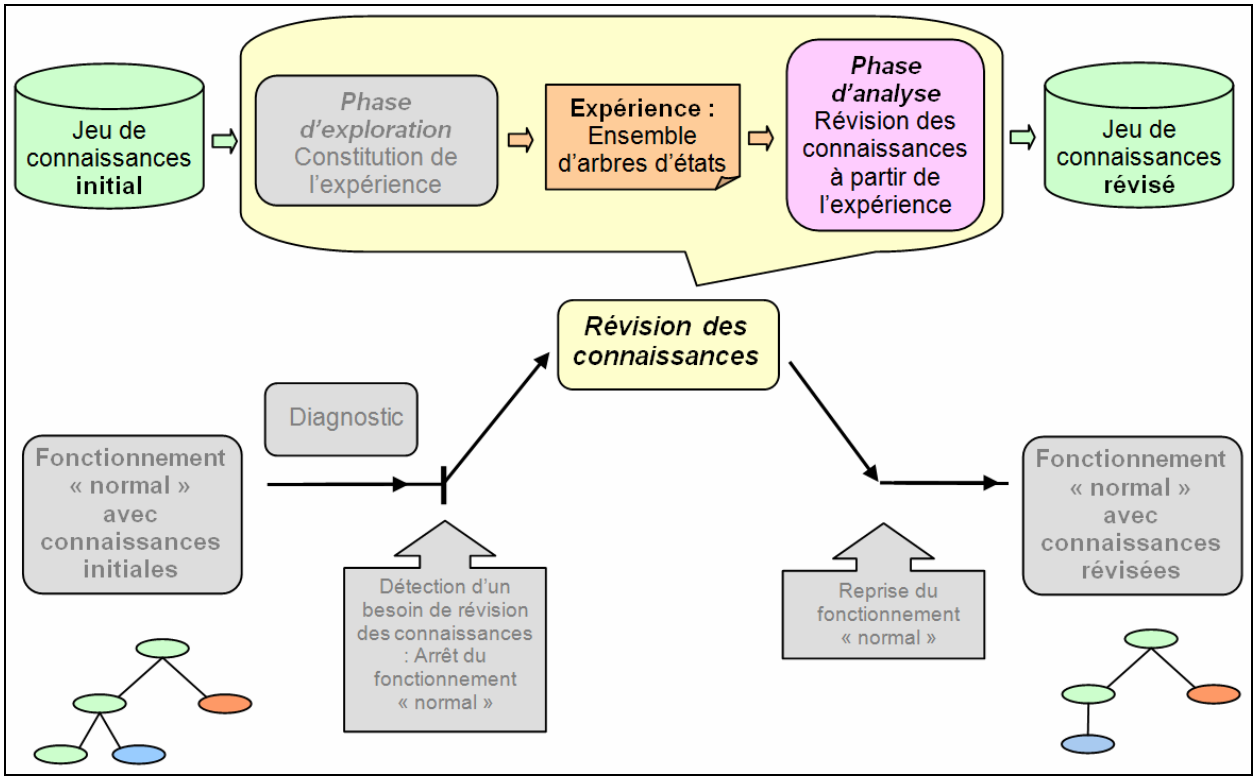


Figure D.1 Phase d'analyse

Nous présenterons, dans une première partie (D.II), les problématiques liées au *processus de révision par analyse* ainsi que l'approche générale que nous proposons pour ce processus. Notre approche se base sur la décomposition de l'ensemble des connaissances en groupes de connaissances et sur le déclenchement séquentiel d'un *sous-processus de révision par analyse* par groupe de connaissances.

Nous nous intéresserons en partie D.III à ces *sous-processus de révision par analyse*. Nous présenterons ainsi, en partie D.III.1, une approche générique pour ces sous-processus basée sur l'exploration du domaine de définition des connaissances. Nous présenterons, dans ce cadre, un court état de l'art des méthodes d'exploration en intelligence artificielle.
Cette approche générique peut se révéler inefficace dans le cadre de connaissances complexes comme par exemple les connaissances représentées sous forme de bases de règles. Des approches plus spécialisées seront alors nécessaires. En vue de permettre la révision de l'ensemble des connaissances du modèle AGENT, nous proposerons en partie D.III.2 une approche de *révision par analyse* destinée aux connaissances représentées sous forme de bases de règles de production.

D.II Approche générale de révision par analyse

D.II.1 Problématiques liées au processus de révision par analyse

L'objectif de notre processus général de révision par analyse est d'améliorer, si besoin, l'ensemble des connaissances du système de résolution de problèmes, c'est-à-dire de déterminer pour chaque connaissance, quelle valeur de son domaine de définition choisir afin de rendre le système le plus performant possible pour l'ensemble des instances possibles du problème d'optimisation considéré.

Une première difficulté relative à ce processus provient de la combinatoire très importante. En effet, le nombre de connaissances procédurales utilisées pour la résolution d'un problème d'optimisation peut être très élevé ainsi que la taille du domaine de définition de chacune de ces connaissances. Par exemple, le modèle AGENT tel que nous l'avons enrichi comprend de nombreuses connaissances : pour chaque contrainte, une connaissance de priorité ; pour chaque action, une connaissance d'application et une connaissance de restriction ; pour l'agent géographique en lui-même, une connaissance de fin de cycle, une d'optimalité et une de validité des états. Le nombre Nb_K de connaissances procédurales en jeu pour la généralisation d'un type d'agent géographique est donc de :

$$Nb_K = Nombre\ de\ contraintes \times ((Nombre\ d'actions)^2 + 3)$$

Il peut donc s'avérer extrêmement complexe de réviser l'ensemble des connaissances du système de résolution de problèmes en même temps.

Il n'est pas judicieux de chercher à réduire cette combinatoire en révisant les connaissances totalement indépendamment les unes des autres. En effet, comme nous l'avons montré en partie B.III.1, des interdépendances peuvent exister entre elles. Celles-ci peuvent être de deux types : *directes* ou *indirectes*.

Des connaissances sont en interdépendance *directe* si elles ne peuvent être comprises indépendamment les unes des autres. C'est le cas des connaissances sur la priorité des contraintes dans le modèle AGENT. En effet, si l'expression des règles de priorité de chaque contrainte se fait de façon totalement indépendante (chaque contrainte dispose d'une base de règles qui ne dépend que de sa propre satisfaction), le résultat obtenu (la priorité) ne peut être analysé que s'il est comparé aux priorités des autres contraintes de l'agent géographique. En effet, pour un agent géographique, qui a deux contraintes C_1 et C_2, savoir que pour un état donné, la priorité de C_1 vaut 5 n'a en soi pas de sens si on ne sait pas que pour le même état, la priorité de C_2 est égale à 3.

Des connaissances sont en interdépendance *indirecte*, si elles peuvent être comprises indépendamment les unes des autres, et si les résultats en termes de succès et d'échec obtenus par l'une peuvent dépendre des valeurs affectées aux autres connaissances. Par exemple, dans le cadre du modèle AGENT, la connaissance de validité est en interdépendance *indirecte* avec celles d'application des actions : en fonction du critère de validité utilisé, certaines actions peuvent mener soit à un succès, soit à un échec.

Définir un processus général d'analyse demande donc de définir une décomposition de l'ensemble des connaissances en groupes de connaissances (pouvant contenir une ou plusieurs connaissances) permettant à la fois de limiter les problèmes d'interdépendance et d'éviter une explosion combinatoire.

Une seconde question posée est de savoir, une fois la décomposition de l'ensemble des connaissances en groupes de connaissances accomplie, comment réviser chaque groupe à partir des traces obtenues durant la phase d'exploration.

Enfin, une dernière question déjà posée en partie B.III.3 concerne la prise en compte du jeu de connaissances initial. Nous avons proposé en partie B.V.2 d'intégrer un *coefficient de qualité des connaissances* permettant de définir a priori la qualité d'un jeu de connaissances. Comment intégrer ce coefficient dans le *processus de révision par analyse* afin que celui-ci soit amené, en fonction de la valeur de ce coefficient, à plus ou moins modifier la valeur du jeu de connaissances initial ?

Nous présentons dans la partie suivante (D.II.2) notre approche générale de révision. Cette dernière est basée sur la décomposition du processus général d'analyse en sous-processus et sur la recherche, pour chaque connaissance, de la meilleure valeur de son domaine de définition à lui affecter.

D.II.2 Présentation de l'approche générale de révision par analyse

D.II.2.1 Décomposition du processus général de révision par analyse

D.II.2.1.1 Approche générale

Comme nous l'avons évoqué en partie D.II.1, la combinatoire, c'est-à-dire le nombre de combinaisons différentes de jeux de connaissances possibles, peut s'avérer très importante. Il sera donc souvent indispensable de décomposer l'ensemble des connaissances en groupes de connaissances distincts. Un problème de cette décomposition provient des interdépendances existant entre connaissances. Nous avons défini deux types d'interdépendances : les interdépendances *directes* et *indirectes*. Idéalement, chaque groupe de connaissances devrait être totalement indépendant des autres, c'est-à-dire que chaque connaissance du groupe devrait être totalement indépendante des connaissances des autres groupes. Malheureusement, dans de nombreux systèmes de résolution de problèmes (comme ceux basés sur le modèle AGENT), les connaissances sont toutes interdépendantes. Il est donc nécessaire d'accepter que deux groupes de connaissances puissent partager des interdépendances. Parmi les deux types d'interdépendances définis, les interdépendances *directes* posent plus de problèmes pour le *processus de révision par analyse*. En effet, il est très complexe d'analyser et de réviser une connaissance qui ne peut être comprise que par comparaison à d'autres connaissances.

Nous proposons donc de diviser les connaissances en groupes de connaissances non directement dépendants, c'est-à-dire en groupes dont chaque connaissance ne peut partager d'interdépendances *directes* avec les connaissances des autres groupes, mais peut partager des interdépendances *indirectes*. Dans le cadre du modèle AGENT, les groupes de connaissances non directement dépendants correspondent aux différents types de connaissances : ainsi, les connaissances sont divisées en six groupes : un pour les connaissances sur la priorité des contraintes, un pour celles sur l'application des actions, un pour celles sur la restriction des

actions, un pour celle sur la validité des états, un pour celle sur l'optimalité des états et un pour celle sur la fin de cycle d'actions.

Plusieurs approches peuvent être adoptées pour procéder à l'analyse des différents groupes de connaissances.

La plus simple consiste à procéder à l'analyse des groupes indépendamment les uns des autres. Ce type d'approche est pertinent lorsque les groupes de connaissances sont totalement indépendants les uns des autres, ce qui n'est pas le cas de nombreux systèmes de résolution de problèmes dont ceux basés sur le modèle AGENT. Dans le cas contraire, une telle approche peut entraîner de graves effets de bord.

La figure D.2 présente un exemple de problème pouvant être occasionné par une révision indépendante des connaissances. Cet exemple reprend le système de résolution de problèmes (avec ses deux connaissances) et l'arbre d'états déjà présenté en figure C.3 de la partie C.III.1. Les deux connaissances du système de résolution de problèmes sont indirectement interdépendantes.

Nous avons choisi, pour la révision des connaissances, comme fonction d'évaluation $Perf(S_K,p)$ des performances d'un système de résolution S utilisant un jeu de connaissances K pour la résolution d'une instance p, la fonction :

$$Perf(S_K, p) = 30 \times Q_P(meilleur\_état(p)) - nb\_état(p)$$

Cette fonction permet de privilégier, par le biais d'un facteur 30, les jeux de connaissances assurant une bonne efficacité du système au détriment que ceux assurant une bonne efficience. Le choix d'un facteur 30 a juste valeur d'exemple.

Dans cet exemple les deux connaissances sont révisées indépendamment. Nous révisons donc la connaissance d'application des actions en prenant, comme valeur pour le critère de validité, celle définie dans le jeu de connaissances initial (état valide si Q_P(état courant) $\geq$ Q_P(état précédent)) et de même, nous révisons le critère de validité en prenant, comme valeur pour la connaissance d'application des actions, celle définie dans le jeu de connaissances initial (Toujours proposer Action 1 et Action 2).

Nous pouvons constater dans le cadre 1 de la figure D.2, que parmi les quatre valeurs possibles pour la connaissance d'application des actions, celle qui obtient le meilleur score de performance est la valeur « Toujours proposer Action 1 et jamais Action 2 ». Elle obtient en effet un score de performance de 20. Nous pouvons constater également, dans le cadre 2, que parmi les deux valeurs possibles pour le critère de validité, celle qui obtient le meilleur score de performance avec un score de 17.5 est la valeur « état valide si Q_P(état courant) > Q_P(état précédent) ».

La révision de chacune des deux connaissances a bien permis, pour celles-ci, une augmentation de la valeur de performance du système. Néanmoins, une fois les connaissances révisées réunies, les performances du système se montrent très mauvaises (même moins bonnes qu'elles ne l'étaient avec les connaissances initiales). Réviser indépendamment les connaissances peut donc s'avérer néfaste pour les performances globales du système de résolution de problèmes.

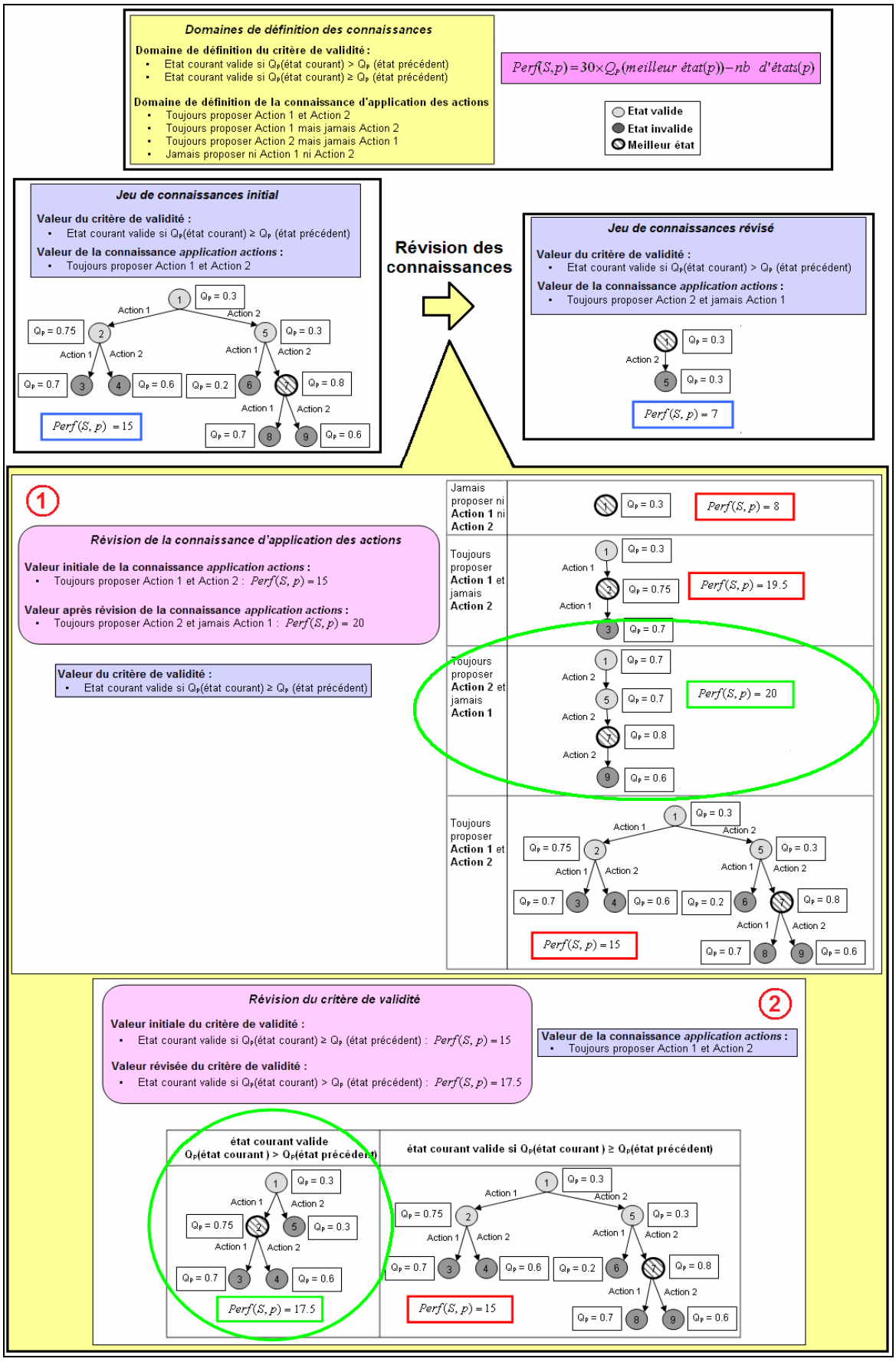


Figure D.2 Problème de la révision indépendante des connaissances

Afin de limiter ces effets de bord, nous proposons de procéder à l'analyse de chaque groupe de connaissances séquentiellement en tenant compte des groupes de connaissances déjà révisées. Nous proposons à cet effet de ne prendre en compte qu'un seul groupe de connaissances initiales à la fois (le groupe à réviser) et d'utiliser pour les autres groupes soit le groupe de connaissances révisées quand il existe, soit, dans le cas contraire, celui provenant du jeu de connaissances *minimal* défini en partie C.III.1. La figure D.3 illustre notre approche.

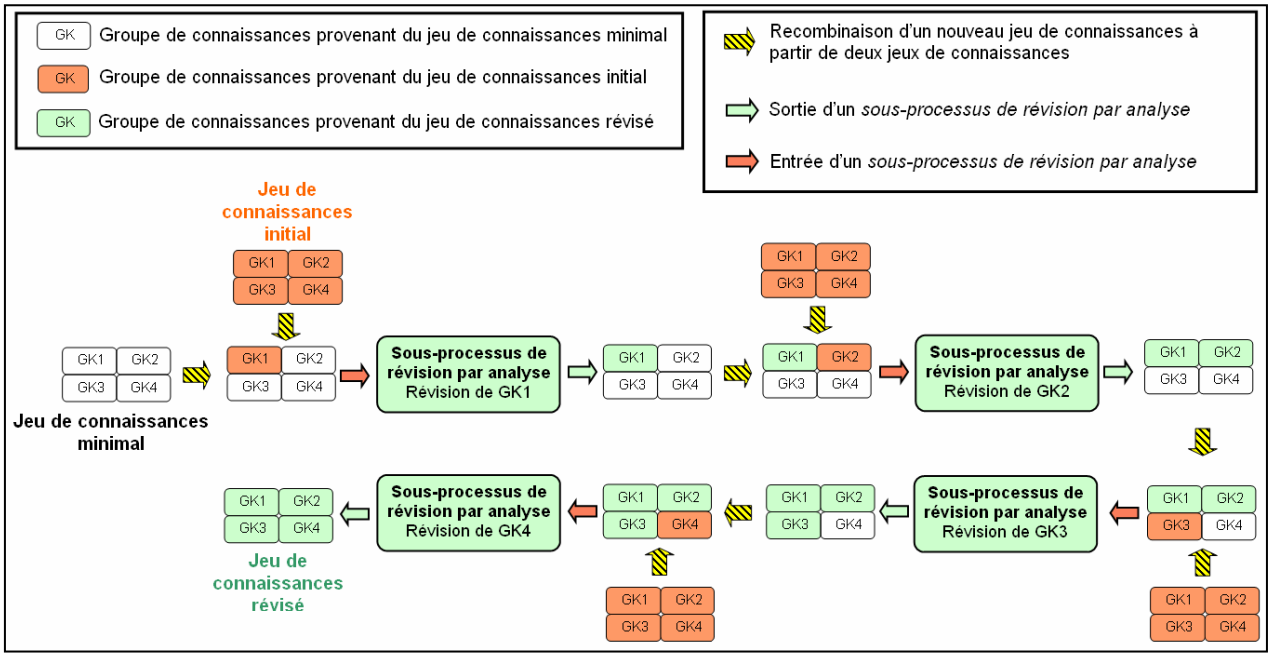


Figure D.3 Approche d'analyse par application séquentielle des *sous-processus de révision par analyse*

L'exemple de la figure D.2 montre qu'il est intéressant de prendre en considération les groupes de connaissances déjà révisées plutôt que les groupes de connaissances initiales. L'avantage de partir des groupes de connaissances tirés du jeu de connaissances *minimal* plutôt que de ceux provenant des connaissances initiales provient des caractéristiques des connaissances *minimales* et plus particulièrement du fait qu'elles permettent d'explorer tous les états possibles (voir partie C.III.1). En effet, contrairement aux connaissances initiales, les connaissances *minimales* n'entraînent pas d'introduction de biais dans la façon d'explorer l'arbre d'états. S'il est intéressant de prendre en compte le biais introduit par les connaissances révisées car celui-ci est définitif et ne sera pas remis en cause durant le processus d'analyse, le biais introduit par les connaissances initiales a de fortes chances de ne plus être valable une fois les connaissances l'introduisant révisées.

Si cette approche avait été utilisée pour l'exemple proposé figure D.2, et si la connaissance d'application des actions avait été révisée avant le critère de validité, le jeu de connaissances obtenu aurait été le troisième jeu du cadre 1 (celui avec un score de performance de 20). En effet, si la révision de la connaissance d'application des actions s'était déroulée exactement de la même manière que dans le cadre 1 (la valeur obtenue après révision pour la connaissance d'application des actions est la valeur « Toujours proposer Action 1 et jamais Action 2 »), la révision du critère de validité aurait pris en compte la valeur obtenue après révision pour la connaissance d'application des actions et ainsi gardé la valeur initiale du critère de validité (état valide si Q_P(état courant) ≥ Q_P(état précédent)).

Dans le cas où le critère de validité aurait été révisé avant la connaissance d'application des actions, le jeu de connaissances obtenu aurait été le 3 de la figure C.2, à savoir un jeu de connaissances obtenant un score de performance de 19,5.

Cet exemple illustre l'intérêt de cette approche de révision séquentielle des connaissances et montre que le jeu de connaissances obtenu après révision peut dépendre de l'ordre dans lequel les différents groupes de connaissances sont révisés. Si le nombre de fois où chaque connaissance est révisée n'est pas limité, le nombre de combinaisons possibles pour l'enchaînement des *sous-processus de révision par analyse* peut être infini. Il est donc important de disposer d'heuristiques pour limiter ce nombre. Ces dernières doivent être définies en fonction du système et des types de connaissances traités. Nous présentons en partie D.II.2.1.2, les heuristiques que nous proposons pour le modèle AGENT.

D.II.2.1.2 Application pour le modèle AGENT

Les différents groupes de connaissances, issus de la division proposée en partie D.II.2.1.1, sont pour le modèle AGENT les suivants :

- Groupe des connaissances sur la priorité des contraintes
- Groupe des connaissances sur l'application des actions
- Groupe des connaissances sur la restriction des actions
- Groupe de la connaissance sur l'optimalité des états
- Groupe de la connaissance sur la validité des états
- Groupe de la connaissance sur la fin de cycle

A noter que certains groupes de connaissances sont composés de plusieurs connaissances (comme c'est le cas pour le groupe de connaissances sur la priorité des contraintes) alors que d'autres sont composés d'une unique connaissance (comme c'est le cas pour le groupe de la connaissance sur la validité des états).

Nous posons comme première contrainte pour le séquençage des processus d'analyse que chaque groupe de connaissances n'est révisé qu'une seule fois. Cette contrainte forte permet de passer d'un nombre infini de séquences possibles pour l'enchaînement des *sous-processus de révision par analyse* à un nombre fini. Nous proposerons en partie F, une expérimentation sur l'impact de cette contrainte sur les résultats de révision pouvant être obtenus.

Le nombre de séquences possibles pour l'enchaînement des *sous-processus de révision par analyse* est donc de 6!, soit 720 séquences possibles. En fonction des besoins de l'utilisateur et du temps nécessaire à l'exécution des *sous-processus de révision par analyse*, ce nombre peut s'avérer trop important pour pouvoir permettre de procéder à un test exhaustif de l'ensemble des ordres possibles. Pour donner un ordre d'idées, un temps réaliste compte tenu des expérimentations pour une séquence complète de *sous-processus de révision par analyse* est de 5 minutes, ce qui entraîne donc un temps de 60 heures pour pouvoir tester l'ensemble des séquences possibles.

Nous proposons donc, dans le cadre de cette thèse, de nous limiter à un ordre particulier. Notre idée générale est d'essayer, dans un premier temps, de guider du mieux possible l'exploration (en mettant les meilleurs états le plus à gauche possible des arbres) puis de chercher à élaguer ces arbres. Nous proposons donc de réviser, en premier lieu, les groupes de connaissances relatifs au guidage du processus (connaissances relatives à la priorité des contraintes et à l'application des actions) puis de réviser ceux relatifs à l'élagage des arbres

(connaissances relatives à la validité des états, à leur optimalité, à la fin de cycle et aux restrictions sur les actions).

Concernant nos deux types de connaissances de guidage, nous pouvons considérer que les connaissances relatives à la priorité des contraintes auront un impact plus important sur le guidage que celles relatives à l'application des actions (le choix des actions à appliquer en priorité est d'abord dépendant des priorités des contraintes). Nous proposons donc de réviser le groupe de connaissances sur la priorité des contraintes en priorité.

Concernant les quatre autres types de connaissances, nous constatons que :

- La connaissance relative à la validité des états est directement dépendante de la façon dont est parcouru l'arbre d'états et introduit un élagage qui peut être dépendant de l'ensemble des états parcourus. Réviser les autres connaissances en tenant compte du critère de validité risque donc de biaiser fortement les informations pouvant être tirées des objets géographiques généralisés. Nous proposons donc de lancer le processus de révision de la connaissance de validité des états après avoir révisé l'ensemble des autres connaissances afin de limiter les biais que cette connaissance peut apporter.
- Les connaissances relatives aux restrictions d'application des actions dépendent de la séquence d'actions appliquée avant d'arriver à un état courant. Ces connaissances peuvent donc elles aussi entraîner un biais dans la révision des autres connaissances. Nous proposons ainsi de réviser ce groupe de connaissances après avoir révisé les autres connaissances (à l'exception de la connaissance sur la validité des états).
- Une erreur de la connaissance relative à la fin de cycle peut avoir des conséquences plus graves qu'une erreur de la connaissance relative à l'optimalité des états. Nous préférons donc chercher en priorité à élaguer les arbres d'états par des règles d'optimalité plutôt que par des règles de fin de cycle. Nous proposons donc de réviser en priorité les règles d'optimalité.

Nous proposons donc l'ordre de révision suivant :

1. Révision du groupe des connaissances sur la priorité des contraintes
2. Révision du groupe des connaissances sur l'application des actions
3. Révision du groupe de la connaissance sur l'optimalité des états
4. Révision du groupe de la connaissance sur la fin de cycle
5. Révision du groupe des connaissances sur la restriction des actions
6. Révision du groupe de la connaissance sur la validité des états

D.II.2.2 Sous-processus de révision par analyse pour un groupe de connaissances

L'approche générale d'analyse que nous avons présentée permet de définir comment orchestrer les *sous-processus de révision par analyse* qui seront appliqués sur les différents groupes de connaissances. Nous présentons dans cette partie D.II.2.2, notre approche générale de révision par analyse pour un groupe de connaissances.

Notre approche de révision par analyse se base sur l'analyse des traces. Ces dernières sont construites durant la phase d'exploration et permettent d'obtenir des informations relatives aux performances globales du système pour un jeu de connaissances donné par simulation de ce dernier sur l'échantillon de révision.

Nous avons évoqué en partie C.IV le fait qu'il était possible de construire, pour les connaissances qui dépendent de l'état courant de l'entité considérée, des informations locales relatives aux succès et aux échecs obtenus par ces connaissances sur l'échantillon de révision (les bases d'exemples). Nous proposons d'utiliser également ces bases d'exemples pour certains *sous-processus de révision par analyse*.

Notre approche générale d'analyse est composée de deux étapes :

- La première étape consiste, pour les connaissances qui dépendent de l'état courant de l'entité considérée, à analyser les traces construites durant la phase d'exploration pour en tirer des informations relatives aux succès et aux échecs des connaissances et ainsi construire des bases d'exemples (cf. C.IV).
- La seconde étape consiste à tirer parti des bases d'exemples ainsi que des informations sur les performances globales du système pour réviser les connaissances du groupe considéré (cf. D.II.2.2.3). L'objectif de cette étape est de déterminer comment modifier la valeur des connaissances initiales afin qu'elles maximisent une fonction d'évaluation. Nous revenons plus en détail sur cette étape de révision des connaissances en partie D.III.

D.III Révision par analyse d'un groupe de connaissances

D.III.1 Introduction

Nous avons présenté, en partie D.II.2.1, notre approche de *révision par analyse*. Nous avons ainsi proposé de diviser le processus complet de *révision par analyse* en sous-processus activés séquentiellement. Chaque sous-processus est chargé de réviser un groupe de connaissances.

Nous avons ensuite proposé, en partie D.II.2.2, une approche générale de *révision par analyse* d'un groupe de connaissances. Cette approche générale consiste à tirer parti d'informations globales et éventuellement d'informations locales afin de définir la meilleure valeur à affecter à chacune des connaissances du groupe.

Nous proposons dans ce cadre, en partie D.III.2, une approche générique pouvant être utilisée pour tout groupe de connaissances. Cette approche est basée sur l'exploration du domaine de définition de chacune des connaissances à l'aide de techniques de résolution de problèmes. Nous présenterons donc un court état de l'art des méthodes de résolution de problèmes en intelligence artificielle.

Notre approche générique peut se révéler peu performante pour les connaissances complexes. Ainsi cette approche se montrera peu adaptée pour les connaissances représentées sous forme de bases de règles. Nous proposerons donc, en partie D.III.3, une approche spécialisée pour ce type de connaissances, basée sur la réduction des domaines de définition des connaissances et sur l'exploration de ces domaines réduits.

D.III.2 Approche générique de révision par analyse d'un groupe de connaissances

D.III.2.1 Formalisation du problème considéré et approche proposée

Nous proposons, dans cette partie D.III.2, une approche générique de révision par analyse d'un groupe de connaissances. Cette approche est utilisable quel que soit le formalisme de représentation des connaissances composant le groupe.
Nous ne pourrons donc pas utiliser les bases d'exemples pour guider le processus de révision car celles-ci ne peuvent être construites que pour des connaissances d'un type particulier (celles qui dépendent uniquement de l'état courant de l'entité considérée).

Nous proposons de formaliser le problème de la révision par analyse d'un groupe de connaissances sous la forme d'un problème de recherche d'un jeu de connaissances maximisant une certaine fonction d'évaluation.

Soit un groupe de connaissances G_K composé d'un ensemble de connaissances $\{K_i\}$ admettant chacune un domaine de définition $\{valeur\}_{Ki}$. L'objectif de la révision par analyse de ce groupe de connaissances consiste à affecter à chaque K_i une valeur permettant de maximiser la fonction d'évaluation $Eval(K_{init}, G_K, S_{Krévisée}, P_n)$. Cette fonction d'évaluation, que nous présentons en partie D.III.2.2, traduit la pertinence du fait de remplacer le jeu de

connaissances initial K_{init} par un jeu de connaissances $K_{révisée}$ pour un système *S* et un échantillon d'instances de problème *Pn*.

On se retrouve face à un problème classique d'affectation de valeur dont la taille de l'espace de recherche est de $\prod_{K_i \in G_K} card(\{valeur\}_{K_i})$

En fonction des groupes de connaissances considérés, l'espace peut s'avérer trop important pour permettre un test exhaustif de l'ensemble des combinaisons possibles d'attributions de valeurs (jeux de connaissances). Dans ce cadre, l'utilisation de techniques d'exploration incomplète est indispensable. Nous proposons un court état de l'art de ces techniques en partie D.III.2.3.

Notre approche générique consiste donc à explorer l'espace des domaines de définition de l'ensemble des connaissances à l'aide de techniques d'exploration afin de trouver un jeu de connaissances maximisant la fonction d'évaluation $Eval(K_{init}, G_K, S_{Krévisée}, P_n)$.

La partie D.III.2.4 est consacrée à l'application de notre approche générique pour les groupes de connaissances du modèle AGENT constitués, pour le premier, de la connaissance sur la validité des états, et pour le second, des connaissances sur la restriction d'application des actions. En effet, les domaines de définition de ces deux connaissances étant très restreints, le coût, en termes de temps de calcul, du test exhaustif de l'ensemble des combinaisons possibles est très faible. Notre approche générique est donc très adaptée à la révision de ces deux groupes de connaissances.

D.III.2.2 Fonction d'évaluation de la pertinence du remplacement du jeu de connaissances initial par un nouveau jeu de connaissances

Nous avons introduit au chapitre B.V.2, un *coefficient de qualité des connaissances* représentant, pour chaque connaissance, sa qualité a priori et traduisant le taux de modifications autorisé lors de sa révision. Ainsi, plus ce coefficient est faible pour une connaissance, plus celle-ci pourra être modifiée lors du déclenchement du processus de révision.

Nous proposons donc, afin de tenir compte de ce coefficient et de limiter les modifications des connaissances qui ont une valeur élevée pour celui-ci, de faire intervenir celui-ci dans le calcul de l'évaluation d'un jeu de connaissances. Nous définissons ainsi une fonction permettant d'évaluer s'il est ou non pertinent, pour un groupe de connaissances, de remplacer le jeu de connaissances initial K_{init} par un nouveau jeu de connaissances $K_{révisé}$. Nous rappelons qu'un groupe de connaissances est un ensemble d'au moins une connaissance où les connaissances sont en interdépendance *directe* entre elles.

Soient un système *S*, un échantillon de révision P_n, un groupe de connaissances G_K, un jeu de connaissances initial K_{init} et un jeu de connaissances révisé $K_{révisé}$. Nous définissons la fonction d'évaluation, $Eval(K_{init}, G_K, S_{Krévisé}, P_n)$, qui évalue la pertinence du remplacement du jeu de connaissances initial K_{init} par un nouveau jeu de connaissances $K_{révisé}$, par :

$$Eval(K_{init}, G_K, S_{K_{révisée}}, P_n) = TauxAcceptModif(G_K, K_{init}, K_{révisé}) \times Perf(S_{K_{révisée}}, P_n)$$

La fonction $Perf(S_{K_{révisé}}, P_n)$ est la fonction d'évaluation des performances d'un système *S* utilisant un jeu de connaissances $K_{révisé}$ pour un échantillon de révision P_n (cf. B.III.2.1). Nous rappelons d'ailleurs que l'utilisation *d'un jeu de connaissances minimal* durant la phase d'exploration nous permet de ne pas avoir, pour le test d'un jeu de connaissances pour la résolution d'une instance d'un problème, à reconstruire l'ensemble des états mais juste à réarranger les états déjà construits (C.III.1). Ainsi, le temps de calcul du test d'un jeu de connaissances pour la résolution d'une instance d'un problème et donc pour le calcul de la fonction $Perf(S_{K_{révisé}}, P_n)$ est extrêmement court.

TxAcceptModif(G_K, K_{init}, $K_{révisé}$) est une fonction à valeurs réelles définies sur [0,1], qui traduit le taux d'acceptation des modifications. Une valeur de 0 pour cette fonction signifie que les connaissances ont connu trop de modifications par rapport aux modifications acceptées et une valeur de 1 signifie que les modifications restent dans le cadre des modifications admises.

L'intérêt d'introduire une telle fonction dans le calcul de l'évaluation d'un jeu de connaissances est de limiter les modifications trop importantes des connaissances ayant une valeur de coefficient de qualité élevée, mais, en même temps, de permettre des modifications du jeu de connaissances s'il s'avère possible d'améliorer significativement la qualité de celui-ci.

Soit un jeu de connaissances *K*, on note $Val_i(K)$, la valeur de la connaissance *i* pour le jeu de connaissances *K*. On note de même, *CoeffQK(i)*, la valeur du coefficient de qualité affectée à *l'agent connaissance* représentant la connaissance *i*. Nous rappelons que ce coefficient compris entre 0 et 1 définit la qualité a priori de la connaissance et donc si celle-ci doit ou non être modifiée durant le processus de révision (cf. B.V.2).

Nous proposons de définir la fonction *TxAcceptModif*(G_K, K_{init}, $K_{révisé}$) par :

$$TxAcceptModif(G_K, K_{init}, K_{révisée}) = \underset{i \in G_K}{Min}\left(TxAcceptModif_i(Val_i(K_{init}), Val_i(K_{révisée}))\right)$$

avec

$$TxAcceptModif_i(Val_{init}, Val_{révisée}) = \begin{cases} 1 & \textbf{si } TxModif_i(Val_{init}, Val_{révisée}) = 0 \\ 1 - Min(1, CoeffQK(i) \times (1 + TxModif_i(Val_{init}, Val_{révisée})) & \textbf{si } TxModif_i(Val_{init}, Val_{révisée}) \neq 0 \end{cases}$$

La fonction $TxModif_i(x,y)$ est une fonction à valeurs réelles définies sur [0,1], qui traduit le taux de modification de la valeur d'une connaissance *i*. Une valeur de 0 signifie que les deux valeurs *x* et *y* de *i* sont similaires. Une valeur de 1, signifie que les deux valeurs sont totalement différentes. Le choix de cette fonction dépendra du type de connaissances considéré.

L'utilisation d'une fonction « *Min* » pour la fonction *TxAcceptModif*(G_K, K_{init}, $K_{révisé}$) est justifiée par le fait qu'il est préférable de modifier légèrement la valeur de chaque connaissance que de trop modifier la valeur d'une seule connaissance.

Lorsqu'une connaissance *i* subit de fortes modifications de sa valeur, sa fonction $TxModif_i(x,y)$ tend vers 1. Si *l'agent connaissance* représentant cette connaissance a une

valeur de *coefficient de qualité* égale à 1, la valeur de $TxAcceptModif(G_K, K_{init}, K_{révisé})$ sera égale à 0 et donc la valeur de $Eval(K_{init}, G_K, S_{Krévisé}, P_n)$ sera aussi égale à 0. De ce fait, le jeu de connaissances $K_{révisé}$ ne pourra pas être choisi lors du processus de révision. Cette propriété assure qu'une connaissance dont *l'agent connaissance* qui la représente a un *coefficient de qualité* égal à 1 ne soit jamais modifiée (cf.B.V.2).

Si, la valeur du *coefficient de qualité* de chaque connaissance *i* du groupe de connaissances considéré est égal à 0, la valeur de $TxAcceptModif(G_K, K_{init}, K_{révisé})$ sera égale à 1 et donc la valeur de la fonction d' évaluation sera égale à la valeur de la fonction de performance : si le jeu de connaissances permet une amélioration des performances, celui-ci aura de bonnes chances d'être choisi, dans le cas contraire, il en aura peu.

Nous avons introduit, dans cette partie D.III.2.2, une fonction permettant d'évaluer la pertinence de modifications apportées au jeu de connaissances initial. Notre approche générique de révision se base sur l'utilisation de méthodes de résolution de problèmes qui permettront d'explorer le domaine de définition de chacune des connaissances du groupe afin de trouver un jeu de connaissances maximisant la fonction d'évaluation. Nous introduisons dans la partie suivante (D.III.2.3) un court état de l'art des méthodes de résolution de problèmes en intelligence artificielle.

D.III.2.3 Méthodes de résolution de problèmes en intelligence artificielle

D.III.2.3.1 Introduction

La résolution de problèmes fait partie des thèmes centraux de l'intelligence artificielle depuis sa création. Le « Logic Theorist » de Simon et Newell (1956), destiné à démontrer des théorèmes mathématiques par exploration informée d'arbres d'états, est souvent considéré comme le premier programme de la discipline. Depuis ce premier programme, de nombreux travaux ont été menés dans le domaine permettant le développement d'un grand nombre de méthodes.

Ces méthodes sont caractérisées par diverses caractéristiques telles que :

- La *complétude* : une approche complète certifie de trouver une solution de l'instance du problème lorsqu'il en existe une.
- La *correction* : une approche correcte certifie que toute réponse trouvée est solution de l'instance du problème.
- *Informée/non informée* : une approche informée utilise des informations spécifiques au problème pour résoudre des instances de celui-ci. Une approche non informée (ou aveugle) ne dispose que de la définition du problème comme information.
- *Stochastique/déterministe* : une approche déterministe, au contraire d'une approche stochastique, renvoie toujours les mêmes réponses pour une même instance d'un problème.

La question du choix d'une approche est souvent guidée par les caractéristiques du problème à résoudre. Dans le cadre de nos travaux où la combinatoire sera généralement très importante (la taille de l'espace de recherche), nous privilégierons les approches incomplètes. De même, nous souhaitons que les réponses proposées par la méthode de résolution de problèmes soient forcément des solutions de l'instance du problème. Nous ne présenterons donc dans la suite de cet état de l'art que des approches correctes et incomplètes.

La partie suivante (D.II.2.2.3.2) est ainsi consacrée à la présentation des métaheuristiques les plus classiquement utilisées en résolution de problèmes.

D.III.2.3.2 Métaheuristiques pour la résolution de problèmes

Il existe de nombreuses définitions du concept de *métaheuristique*. Nous proposons ici la définition donnée par (Metaheuristics Network, 2000) :
« A **metaheuristic** is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem. »

Une métaheuristique est donc une stratégie permettant de guider la recherche d'une solution optimale, qui peut être appliquée avec peu de modifications à différents problèmes. La solution finale renvoyée par une métaheuristique ne sera pas forcément optimale en raison du caractère incomplet de l'approche : les approches à base de métaheuristiques ne visitent en général qu'une très petite partie de l'espace des solutions.

De nombreuses classifications des métaheuristiques ont été proposées (Birattari et al., 2001 ; Collette et Siarry, 2003). Nous reprendrons, pour notre part, celle proposée par (Blum & Rolli, 2003) qui divise les métaheuristiques en deux grands groupes : celles basées sur des trajectoires et celles basées sur une population.

D.III.2.3.2.1 Métaheuristiques basées sur des trajectoires

Les métaheuristiques basées sur des trajectoires fonctionnent ainsi : elles partent d'une solution initiale et cherchent à l'améliorer par déplacements successifs dans l'espace des solutions. La dynamique du système résulte de la stratégie d'exploration donnée par la métaheuristique, de la façon dont le problème est représenté, ainsi que de l'instance du problème.

Les algorithmes de recherche locale font partie de ce type de métaheuristiques. Ces algorithmes se basent sur une structuration de l'espace des solutions en termes de voisinage. Soit $N_P(s)$ la fonction renvoyant le voisinage d'une solution s pour un problème P donné. La définition de N_P dépend de P et peut avoir une grande influence sur la qualité des résultats pouvant être obtenus par recherche locale.

La méthode de recherche locale la plus simple est la *recherche locale gloutonne* (appelée aussi *hill-climbing*, *recherche itérative*, *exploration par escalade* ou *descente de gradient*). Soit un problème P consistant à trouver parmi un ensemble S de solutions, celle maximisant une fonction f. Le principe de la recherche locale gloutonne pour la résolution d'une instance de P est le suivant :

- Choix d'une solution $s_0 \in S$
- Déterminer $s' \in N_P(s_0)$ telle que $\forall s \in N_P(s_0), f(s') \geq f(s)$
- Si $f(s') > f(s_0)$, alors $s_0 \leftarrow s'$ et itérer, sinon arrêter l'exploration

Le défaut de cette méthode est qu'elle s'arrête au premier maximum local. Afin de faire face à ce problème, des méthodes proposent, sous certaines conditions, d'autoriser des détériorations de la qualité des solutions au cours de la recherche.

Un exemple de ce type de méthodes est le *recuit simulé* qui permet, en fonction d'une probabilité dépendant d'un paramètre (la température) variant au cours des itérations, d'autoriser des détériorations de la qualité de la solution courante. Cette métaheuristique a été proposée indépendamment par (Kirkpatrick et al., 1983*)* et par (Cerny 1985). Elle est inspirée d'un processus utilisé en métallurgie duquel elle tire son nom. En début de recherche, la température est élevée, permettant ainsi d'effectuer des mouvements détériorant la solution et donc de diversifier la recherche, mais plus la température diminue avec les itérations, moins les détériorations sont autorisées, ce qui permet au processus de recherche de converger vers un optimum. Cette métaheuristique, qui a été appliquée dans de nombreux domaines, est parfois utilisée comme composant de métaheuristiques plus complexes (Blum & Roli, 2003).

Une autre métaheuristique permettant des détériorations de la qualité de la solution est la *recherche taboue* (ou *recherche avec tabou*). Cette métaheuristique introduite par (Glover, 1986) est l'une des plus utilisées en résolution de problèmes. Son principe consiste à mémoriser les derniers mouvements effectués dans une liste (la liste taboue) et d'interdire au processus de recherche d'effectuer des mouvements inverses à ces derniers. Cette interdiction permet de limiter le risque de blocage au niveau d'un maximum local ainsi que les risques de cycle. La liste taboue est de type file (premier entré, premier sorti), seuls les mouvements les plus récents sont conservés. La taille de la liste taboue a une grande influence sur la recherche. Certains travaux ont ainsi proposé de faire varier la taille de la liste taboue en fonction des résultats de la recherche (Battiti & Tecchiolli, 1994).

Il existe de nombreuses autres métaheuristiques basées sur des recherches locales. Nous pouvons ainsi citer *GRASP* (Feo & Resende, 1992) qui est basée sur des successions de phases de construction d'une solution (par une méthode de construction *gloutonne aléatoire*) et de phases d'optimisation de la solution par recherche locale. Une autre métaheuristique très employée est la *recherche à voisinage variable* (Hansen & Mladenovic, 1997). Cette dernière propose de modifier dynamiquement le voisinage en fonction des résultats de la recherche. D'autres métaheuristiques basées sur des recherches locales peuvent être trouvées dans (Blum & Roli, 2003).

D.III.2.3.2.2 Métaheuristiques basées sur une population

Les métaheuristiques de ce type ne prennent plus une seule solution en considération, mais une population de solutions. Leurs performances sont directement liées à la façon dont la population est manipulée. Les deux principales approches de ce domaine sont les algorithmes évolutionnaires et les méthodes d'optimisation par colonie de fourmis.

Les algorithmes évolutionnaires s'inspirent des travaux de Darwin sur la sélection naturelle. Ils appliquent des principes d'évolution à une population de solutions. L'origine de ces algorithmes provient des travaux de (Fogel et al., 1966 ; Holland, 1975). Il existe différents types d'algorithmes évolutionnaires. Les plus célèbres sont les algorithmes génétiques (Goldberg, 1989).

Le principe des algorithmes génétiques est de chercher une solution optimale, par application sur une population initiale de solutions, d'un processus évolutif dirigé par le biais d'opérateurs d'évolution.

Ces opérateurs sont de trois types :

- *Croissement (crossing-over*) : deux solutions sont combinées afin d'en produire une nouvelle.
- *Mutation* : une solution est modifiée.
- *Sélection* : seule une partie de la population est conservée. Différentes techniques peuvent être employées pour effectuer cette sélection. La plupart de celles-ci se basent sur la qualité des solutions.

Il existe une littérature très abondante sur les algorithmes évolutionnaires. Une introduction à ces derniers peut être trouvée dans (Baeck et al., 2003).

D'autres algorithmes de résolution de problèmes s'inspirent également du vivant. Ainsi la métaheuristique ACO (Ant Colony Optimization) trouve ses inspirations dans le fourragement effectué par les colonies de fourmis (Dorigo & Stutzle, 2004). Le principe du fourragement est le suivant : les fourmis qui trouvent de la nourriture, déposent des phéromones sur le chemin de retour. Cette phéromone permet d'attirer d'autres fourmis qui vont elles-mêmes déposer des phéromones sur leur chemin de retour pour attirer d'autres fourmis et ainsi de suite.

Le principe de la métaheuristique ACO est de formaliser le problème à résoudre sous la forme d'un problème de recherche d'un plus court chemin dans un graphe et à faire résoudre ce problème de plus court chemin par des fourmis parcourant ce graphe. Chaque fourmi construit une solution du problème en fonction d'une heuristique propre au problème (qui est la même pour toutes les fourmis) ainsi que des éventuelles traces de phéromones déposées par les autres fourmis. En effet, une fois qu'une fourmi a fini de construire une solution (en parcourant le graphe), elle dépose, en fonction de la qualité de la solution, des phéromones sur les différents nœuds qu'elle a visités. Les fourmis suivantes auront plus de chances de visiter un nœud sur lequel beaucoup de phéromones ont été disposées. Un processus d'évaporation de la phéromone permet "d'oublier" des chemins qui ne sont pas bons. Cette métaheuristique a été utilisée avec succès dans de nombreux domaines (Bullnheimer et al., 1997 ; Gambardella et al., 1999 ; Solnon & Fenet, 2004).

Une stratégie de résolution de problèmes employée par beaucoup d'approches récentes consiste à combiner des métaheuristiques basées sur des populations avec des métaheuristiques basées sur des trajectoires. Par exemple, (Hagemana et al., 2003) proposent de combiner une recherche taboue avec un algorithme génétique.

D.III.2.3.2.3 Méthodes agents pour la résolution de problèmes

La métaheuristique ACO est capable de résoudre un problème par interaction de nombreuses entités extrêmement simples (les fourmis). Ces interactions sont réalisées par l'intermédiaire de la phéromone déposée. Il est possible de considérer ACO comme une approche de résolution par agents réactifs où chaque fourmi est un agent.

Il existe d'autres approches cherchant à résoudre un problème complexe par une "*agentification*" du problème. Un exemple classique de ce type d'approche est l'*éco-résolution* (Drogoul et al, 1991 ; Drogoul, 1993) qui permet de résoudre des problèmes complexes par interaction d'agents réactifs. Chaque agent cherche à se satisfaire et peut être contraint à fuir son état courant sous la pression des autres agents.

Les approches à base d'agents se montrent, en général, très adaptées pour la résolution de problèmes dynamiques et distribués (Smith & Davis, 1981; De Loor et al., 2003). Néanmoins, beaucoup de ces approches, comme (Castro & Oliveira, 2007), sont dédiées à des problèmes particuliers et ne peuvent malheureusement pas être facilement adaptées à d'autres problèmes. Il n'est alors plus possible de parler de métaheuristiques.

D.III.2.3.3 Bilan

Nous avons présenté, dans cette partie D.III.2.3, différentes approches de résolution de problèmes. Il n'existe pas de méthodes simples permettant de définir quelle approche est plus adaptée à un type de problème donné. Néanmoins, les contraintes posées sur un problème peuvent amener à restreindre le choix des approches utilisables.

Comme nous l'avons évoqué en partie D.III.2.3.1, nous privilégions dans notre cadre les approches correctes et incomplètes.

Un aspect important de notre problème est que nous disposons d'une solution initiale (le jeu de connaissances initiales) pour nous aider dans la recherche d'une solution optimale. Il est donc particulièrement intéressant, pour notre problème, de recourir aux métaheuristiques basées sur des trajectoires. En effet, le principe de ces méthodes est de partir d'une solution initiale et de chercher à l'améliorer par des déplacements successifs dans l'espace des solutions.

Il est également possible d'utiliser une méthode agent pour notre problème. En effet, s'il n'existe pas de méthodes agents facilement adaptables à notre problème, il est possible d'utiliser le paradigme agent pour concevoir une méthode adaptée à lui. L'utilisation d'une telle méthode trouve sa justification dans les cas où la révision d'un groupe de connaissances concerne plusieurs connaissances directement interdépendantes entre elles et peut donc se prêter à une résolution distribuée du problème.

D.III.2.4 Application dans le cadre du modèle AGENT

D.III.2.4.1 Révision de la connaissance sur la validité des états

Nous rappelons que le domaine de définition de la connaissance relative à la validité des états est constitué d'un ensemble prédéfini de six critères de validité possibles (cf. B.V.3.1.2) :
CV = {**CEPAC**, **CEPAC** ET **CEPAG**, **CEPAC** ET **CEENS**, **CEPAC** ET **CEEA**, **CEPAC** ET **CEPAG** ET **CEENS**, **CEPAC** ET **CEPAG** ET **CEEA**}.

Le nombre de critères de validité possibles étant faible, nous proposons de tester exhaustivement l'ensemble des critères de validité possibles.

Notre approche demande de définir, pour la connaissance, une fonction *TxModif(K,,K')* qui traduit le taux de modification de la valeur de la connaissance entre le jeu de connaissances *K* et le jeu de connaissances *K'* (cf. D.III.2.2). Nous proposons de définir cette fonction en nous basant sur la notion *d'élagage moins sévère ou égal*.

Un jeu de connaissances *K* admet un *élagage moins sévère ou égal* qu'un autre jeu *K*', si et seulement si, quelle que soit l'instance $p \in P$ à traiter, tous les états construits pour résoudre cette instance avec *K'*, ont aussi été construits avec *K* (cf. C.III.1).

Soit K_{cv} un jeu de connaissances dont le critère de validité a pour valeur *cv*.

CEPAC est le critère utilisé par le *jeu de connaissances minimal*. K_{CEPAC} admet donc un élagage moins sévère que K_{cv} avec $cv \in CV$.
Nous pouvons aussi démontrer que $K_{CEPAC\ ET\ CEENS}$ admet un *élagage moins sévère ou égal* que $K_{CEPAC\ ET\ CEPAG\ ET\ CEENS}$ et que $K_{CEPAC\ ET\ CEEA}$ qui admettent eux-mêmes un *élagage moins sévère ou égal* que $K_{CEPAC\ ET\ CEPAG\ ET\ CEEA}$.

Nous proposons donc de classer ces six critères d'élagage dans cinq catégories d'élagage :
Valeur = 1 : élagage très faible : **CEPAC**
Valeur = 2 : élagage faible : **CEPAC** ET **CEENS**
Valeur = 3 : élagage moyen : **CEPAC** ET **CEPAG**, **CEPAC** ET **CEEA**
Valeur = 4 : élagage sévère : **CEPAC** ET **CEPAG** ET **CEENS**
Valeur = 5 : élagage très sévère : **CEPAC** ET **CEPAG** ET **CEEA**

A noter qu'il n'y a pas relation *d'élagage moins sévère ou égal* entre $K_{CEPAC\ ET\ CEPAG}$ et $K_{CEPAC\ ET\ CEENS}$ et entre $K_{CEPAC\ ET\ CEPAG}$ et $K_{CEPAC\ ET\ CEEA}$. Le choix de placer $K_{CEPAC\ ET\ CEPAG}$ dans la catégorie 3 relève de critères empiriques.

Nous notons *nb_catégories*, le nombre de catégories différentes (5 pour nous), *cat(K)*, la fonction qui renvoie la valeur de la catégorie à laquelle appartient le critère de validité affecté au jeu de connaissances *K*, et *nb_critères(Cat)*, la fonction renvoyant le nombre de critères contenus dans la catégorie de valeur *Cat* (toutes nos catégories sont composées d'un seul critère sauf la catégorie 3 qui est composée de 2 critères).

Nous proposons comme fonction pour *TxModif(K,,K')* :

$$TxModif(K,K') = \begin{cases} 0 & \textbf{si} \quad K = K' \\ \dfrac{1}{nb\_critères(cat(K))} \times \dfrac{1}{nb\_catégories - 1} & \textbf{si} \quad K \neq K' \wedge cat(K) = cat(K') \\ \dfrac{|cat(K) - cat(K')|}{nb\_catégories - 1} & \textbf{si} \quad cat(K) \neq cat(K') \end{cases}$$

Cette fonction vaut 0 lorsque le critère de validité n'a pas été modifié et elle vaut 1 dans le cas où l'on a remplacé le critère de validité admettant l'élagage le moins sévère par le critère de validité le plus sévère (ou vice-versa).

D.III.2.4.2 Révision des connaissances sur la restriction d'application des actions

Le modèle AGENT nécessite de définir pour chaque action, un nombre maximal d'applications. Ce nombre est un entier strictement positif compris entre 0 et ACT_MAX (cf. B.V.3.1.2).

Chaque connaissance de restriction d'application d'une action admet donc un domaine de définition fini. Le nombre de combinaisons possibles pour l'ensemble des connaissances sur la restriction des actions est donc de $(ACT\_MAX + 1)^{nb\ d'actions}$.

Il est possible de faire diminuer ce nombre en remarquant que notre processus de révision est basé sur l'analyse d'instances déjà résolues du problème d'optimisation considéré. Or, il est possible de déterminer, pour chaque instance résolue, le nombre maximal d'applications de chaque action (par branche de l'arbre d'états).

Comme notre approche d'analyse consiste à simuler les différents jeux de connaissances possibles sur l'échantillon de révision, il est inutile de tester les jeux de connaissances proposant, pour une action, une restriction d'application supérieure au nombre maximal d'applications rencontré dans l'échantillon de révision.

Nous proposons donc, pour chaque connaissance de restriction du nombre d'utilisations d'une action, tous les jeux de connaissances proposant pour chaque action *act*, une valeur pour la connaissance relative à la restriction d'application de *act* comprise entre 0 et le *max* entre la valeur maximale d'applications trouvée dans l'échantillon d'apprentissage et la valeur initiale de la restriction.

Soit un agent géographique disposant d'un ensemble d'actions *Act*. Nous notons $nb(act, P_{rev})$, la fonction renvoyant le nombre maximal d'applications d'une action *act* sur un échantillon d'instances de problème d'optimisation $P_{rev,}$ et $valeur\_init(act)$ la fonction renvoyant la valeur initiale de la connaissance sur la restriction d'application de *act*.

Le nombre de jeu de connaissances à tester est donc égal à :

$$nb\_jeux\_K\_à\_tester(Act, P_{rev}) = \prod_{act \in Act} (1 + Max(nb(act, P_{rev}), valeur\_init(act))$$

Le nombre d'actions utilisées par un agent géographique est généralement de l'ordre de 5. Nous avons pu constater empiriquement que le nombre maximal d'utilisations d'une action dépasse rarement 4. Le nombre maximal de jeux de connaissances à tester avec ces données est donc de 3125. Ce nombre est suffisamment faible pour permettre un test exhaustif de l'ensemble des jeux de connaissances possibles. Afin de trouver le meilleur jeu de connaissances possible, nous proposons, dans le cas d'agents géographiques disposant d'un très grand nombre d'actions, de recourir à des méthodes de recherche locale telles que celles présentées en partie D.III.2.3.2.1.

Concernant le choix de la fonction $TxModif(K,K')$ pour cette connaissance, nous proposons la fonction suivante :

$$TxModif(K, K') = \frac{|valeur(K) - valeur(K')|}{Max(valeur(K), valeur(K'))}$$

Ainsi, plus l'écart entre la valeur initiale et la valeur proposée est importante, plus la valeur de la fonction sera proche de 1. Dans le cas où la valeur proposée est égale à la valeur initiale, cette fonction renverra 0.

D.III.3 Approche de révision par analyse pour les connaissances représentées sous forme de bases de règles de production

D.III.3.1Introduction

Nous avons introduit, en partie D.III.2, une approche générique de *révision par analyse d'un groupe de connaissances* qui peut être appliquée à n'importe quel type de connaissances. Cette approche générique peut s'avérer inefficace pour les connaissances complexes telles que celles représentées sous forme de base de règles de production.

Nous proposons donc, dans cette partie D.III.3, une approche de révision par analyse dérivée de notre approche générique et spécialisée pour la révision des connaissances représentées sous forme de bases de règles de production.

La représentation des connaissances par des bases de règles de production offre l'avantage d'être facilement interprétable et permet une traduction naturelle des connaissances de l'expert vers la machine. Ainsi, de nombreux systèmes utilisés dans le cadre d'applications réelles et notamment les systèmes experts (Shortliffe, 1976) utilisent ce type de formalisme pour représenter leurs connaissances.

Le système AGENT est un exemple de système employé dans un contexte industriel utilisant ce type de représentation des connaissances.

Nous proposons, en partie D.III.3.2, une formalisation des connaissances que nous allons considérer. Nous présenterons ensuite, en partie D.III.3.3, notre approche générale de révision pour ces connaissances.

D.III.3.2 Formalisation des connaissances considérées et problématique

D.III.3.2.1 Formalisation des règles considérées

Nous nous intéressons, dans le cadre cette partie D.III.3.2, aux connaissances représentées sous la forme d'un type particulier de règles de production d'ordre 0+.
Nous rappelons que les règles de production sont des règles du type :

```
Si (Prémisse) alors (Conclusion)
```

L'ordre 0+ indique que les règles peuvent dépendre de faits symboliques ou réels mais n'intègrent pas de variables. Un exemple de règles de production d'ordre 0+ est :

```
Si (aire_bâtiment < 100 ET type_bâtiment = quelconque) alors (appliquer
action supprimer_bâtiment)
```

Nous posons comme contrainte sur ces règles, que les prémisses de celles-ci soient uniquement composées de *conditions* connectées par des connecteurs de type « ET » :

$$Prémisse = Condition_1 \text{ ET } Condition_2 \text{ ET } \ldots$$

Chacune des *conditions* doit appartenir à l'ensemble de définition suivant :

- (VRAIE)
- $(x = Vref_{eq})$ avec $Vref_{eq} \in \rho$ ou $Vref_{eq} \in$ {Valeur Qualitative} avec card({Valeur Qualitative}) < N avec $N \in \mathbb{N}$
- $(x < Vref)$ avec $Vref \in \rho$
- $(x \leq Vref)$ avec $Vref \in \rho$
- $(x > Vref)$ avec $Vref \in \rho$
- $(x \geq Vref)$ avec $Vref \in \rho$
- $(x > Vref_1)$ ET $(x < Vref_2)$ avec $(Vref_1, Vref_2) \in \rho^2$ et $(Vref_1 < Vref_2)$
- $(x \geq Vref_1)$ ET $(x < Vref_2)$ avec $(Vref_1, Vref_2) \in \rho^2$ et $(Vref_1 < Vref_2)$
- $(x > Vref_1)$ ET $(x \leq Vref_2)$ avec $(Vref_1, Vref_2) \in \rho^2$ et $(Vref_1 < Vref_2)$
- $(x \geq Vref_1)$ ET $(x \leq Vref_2)$ avec $(Vref_1, Vref_2) \in \rho^2$ et $(Vref_1 < Vref_2)$

avec x, la valeur d'une mesure appartenant au jeu de mesures associé à la connaissance. Nous rappelons que nous avons posé, en partie D.III.1, qu'un jeu de mesures est défini par connaissance. Ce sont ces jeux de mesures qui sont utilisés pour définir les conditions des règles.

La condition $(x = Vref_{eq})$ est particulière car elle peut concerner aussi bien une mesure à valeurs quantitatives (dans ce cas $Vref_{eq}$ est un réel) qu'une mesure à valeurs qualitatives (dans ce cas $Vref_{eq}$ est l'une des valeurs pouvant être prises par la mesure).

Nous ne permettons pas la définition de règles intégrant le connecteur « OU ». Néanmoins, de par le fait que nous nous intéressons à des bases de règles, il est possible de transformer une règle intégrant un « OU » en utilisant deux règles intégrant des « ET ».

Soit la règle :
Si (aire_bâtiment > 400 OU type_bâtiment = fonctionnel) alors (ne pas appliquer action supprimer_bâtiment)

Cette règle peut être traduite sous forme de deux règles :
Si (*aire_bâtiment* > 400) alors (ne pas appliquer *supprimer_bâtiment*)
Si (*type_bâtiment* = fonctionnel) alors (ne pas appliquer *supprimer_bâtiment*)

Nous notons $\mathcal{R}$, l'ensemble des règles répondant aux contraintes posées.

D.III.3.2.2 Formalisation des bases de règles considérées

Nous nous intéressons à un type particulier de bases de règles de type $\mathcal{R}$.

Nous posons comme première contrainte sur nos bases de règles, que les prémisses des règles soient deux à deux contradictoires. Ainsi il ne peut y avoir de règles contradictoires.

Une base de règles est donc de la forme :

Si ($Prémisse_1$) alors ($Conclusion_1$)
Si ($Prémisse_2$) alors ($Conclusion_2$)
…
Si ($Prémisse_n$) alors ($Conclusion_3$)

Avec $\forall (i,j) \in [1,n]$, si $i \neq j$, alors $Premisse_i \cap Premisse_j = \varnothing$

Nous posons comme autre contrainte que les bases de règles couvrent l'ensemble des valeurs pouvant être prises par les mesures du jeu de mesures lié à la base de règles. Ainsi, quelles que soient les valeurs prises par les mesures du jeu de mesures lié à la base de règles, une et une seule conclusion est proposée.

Nous notons $\mathcal{BR}$ l'ensemble des bases de règles répondant aux contraintes posées.

D.III.3.2.3 Problématique de la révision de bases de règles

Un point important du processus de révision est de pouvoir garantir qu'après révision, les bases de règles respectent toujours le formalisme initial de représentation des connaissances. Ce qui implique, que les bases de règles de $\mathcal{BR}$ n'admettent aucune contradiction et couvrent l'ensemble des valeurs de leur jeu de mesures associé.

Nous ne pourrons donc pas procéder à des opérations telles que des suppressions ou des ajouts de règles sans, dans un même temps, mettre à jour les autres règles de la base afin que celle-ci réponde toujours aux contraintes imposées.

Nous avons présenté au chapitre B.II.5 différents travaux portant sur l'affinage de bases de règles. Dans le cadre de nos travaux, les bases de règles répondent à des contraintes particulières (bases de règles de $\mathcal{BR}$). De plus, plusieurs bases de règles doivent parfois être révisées simultanément. Enfin, les données recueillies pour la révision des règles peuvent prendre différentes formes (évaluation de la pertinence des modifications apportées aux bases de règles initiales et bases d'exemples). Les approches d'affinage de bases de règles sont donc peu adaptées.

D.III.3.3 Approche de révision par analyse des bases de règles

Nous présentons, dans cette partie, notre approche générale de révision par analyse des groupes de connaissances représentées par des bases de règles.

Notre approche consiste à explorer l'espace de l'ensemble des bases de règles possibles afin de trouver les bases de règles qui maximisent la fonction d'évaluation $Eval(K_{init},G_K,S_{Krévisé},P_n)$. Cette fonction d'évaluation, que nous avons présentée en partie D.III.2.2, traduit la pertinence du fait de remplacer le jeu de connaissances initial K_{init} par un jeu de connaissances $K_{révisé}$ pour un système S et un échantillon d'instances de problème Pn.

En raison du caractère infini du nombre des valeurs possibles pour les mesures sur lesquelles s'appuient les prémisses des règles, l'espace de l'ensemble des bases de règles possibles est également infini. Nous proposons donc, afin de réduire la taille de cet espace à un espace fini, de partitionner l'espace des mesures en zones qui admettent une conclusion a priori homogène. En effet, l'objectif est de définir des zones dans l'espace des mesures pour lesquelles la connaissance considérée se comporte de façon uniforme.

Notre approche s'appuie sur trois étapes (figure D.4) :

- **Partitionnement**, pour chaque base de règles, de l'espace des mesures en zones admettant une conclusion a priori homogène, à l'aide des bases d'exemples.
  La figure D.4 donne un exemple de partitionnement obtenu. Ainsi, l'espace formé par les mesures M_1 et M_2 a été partitionné en quatre zones auxquelles nous avons affecté les conclusions initiales : lorsque la valeur de M_2 est inférieure à 5, la base de règles

initiale propose d'appliquer l'action Act_1, dans le cas contraire elle propose d'appliquer l'action Act_2.

- **Recherche** d'une affectation complète de conclusions qui optimise la fonction de performance d'évaluation $Eval(K_{init}, G_K, S_{Krévisé}, P_n)$.
  Dans notre exemple figure D.4, la partition définie par une valeur de M_2 comprise entre 3 et 5 s'est vue affecter la conclusion « proposer action Act_2 ». Les autres partitions ont conservé leur affectation initiale.
- **Simplification** des bases de règles finales par agrégation de règles.
  Dans notre exemple figure D.4, les deux partitions ayant pour conclusion « proposer action Act_1 » ont fusionné ainsi que les deux partitions ayant pour conclusion « proposer action Act_2 ».

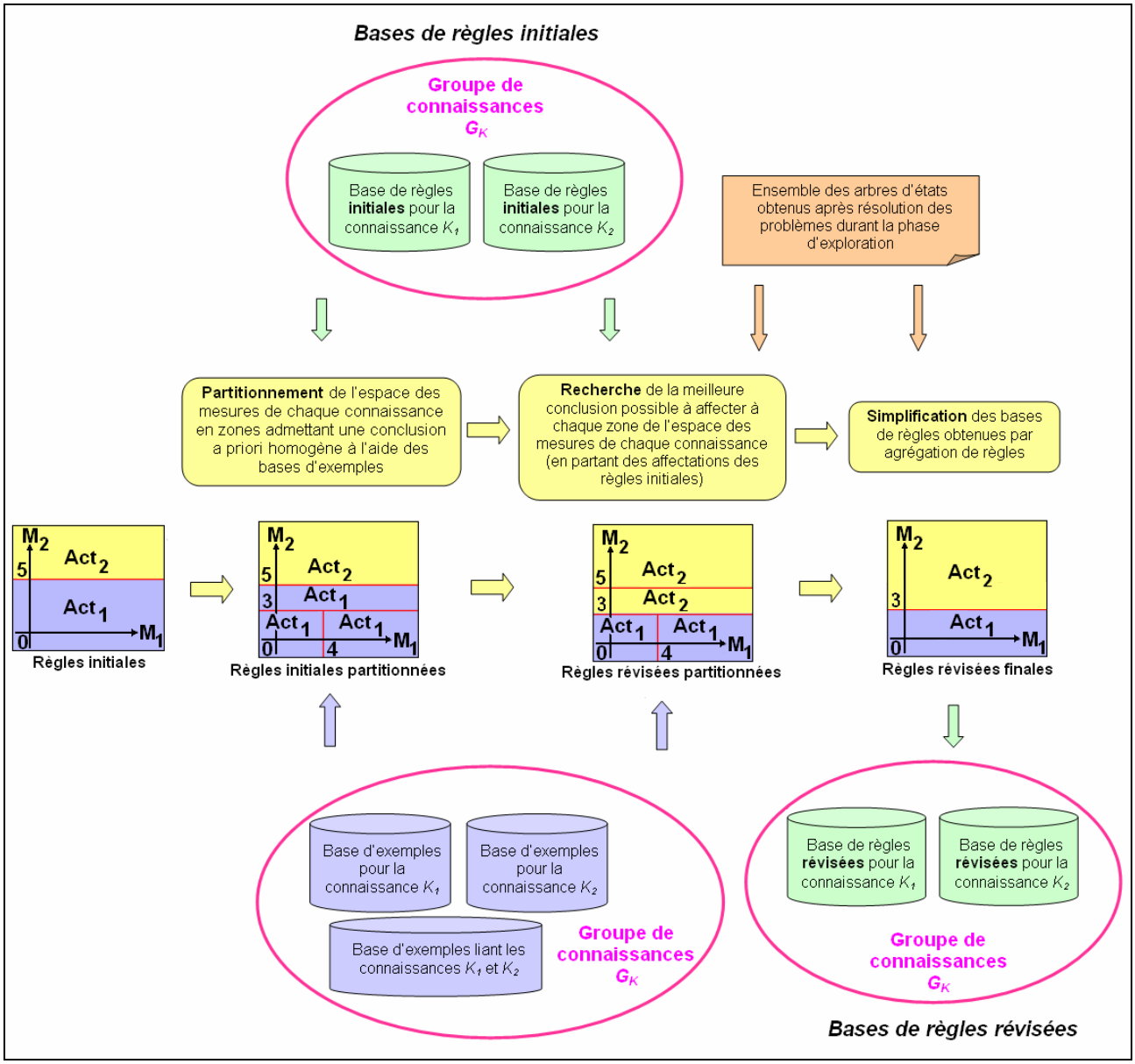


Figure D.4 Approche de révision par partitionnement, puis recherche dans l'espace des règles possibles

Les parties D.III.3.4, D.III.3.5 et D.III.3.6 qui suivent détaillent les trois étapes de cette approche.
Nous revenons, en partie D.III.3.4, sur l'étape de partitionnement de l'espace des mesures. Nous proposons dans ce cadre deux approches de partitionnement ayant chacune des avantages différents.

Nous introduisons, en partie D.III.3.5, différentes approches d'exploration visant à attribuer à chaque zone de l'espace des mesures la meilleure conclusion possible.

Nous proposons enfin, en partie D.III.3.6, une approche de simplification des bases de règles basée sur l'agrégation de règles.

D.III.3.4 Partitionnement de l'espace des mesures

D.III.3.4.1 Introduction

L'objectif de cette étape de partitionnement est de passer, à l'aide de la base d'exemples (exprimée dans un formalisme attributs/valeur), d'un espace de dimension ρ^n a un espace composé d'un nombre fini de partitions admettant chacune une conclusion a priori homogène (figure D.5). Chaque exemple de la base fournit une conclusion pour un point de l'espace des mesures. Il est donc possible, par induction, de passer de ces exemples à un partitionnement de l'espace.

Notre objectif général étant de réviser des connaissances, il est important que le partitionnement tienne compte des connaissances initiales (bases de règles initiales). Nous rappelons que nous avons posé comme première contrainte sur les bases de règles, que celles-ci couvrent l'ensemble des valeurs pouvant être prises par leurs mesures, et comme seconde contrainte que les règles les composant soient disjointes. Nous pouvons donc remarquer, comme illustré sur la figure D.5, que chaque base de règles initiale forme déjà une partition de l'espace de ces mesures où chaque prémisse de règle correspond à une zone de l'espace des mesures. Afin de laisser la possibilité de retrouver ces règles initiales si elles s'avèrent pertinentes, nous proposons de garder ce partitionnement mais de l'affiner en repartitionnant chacune des zones déjà partitionnées en se basant sur les bases d'exemples issues de l'expérience.

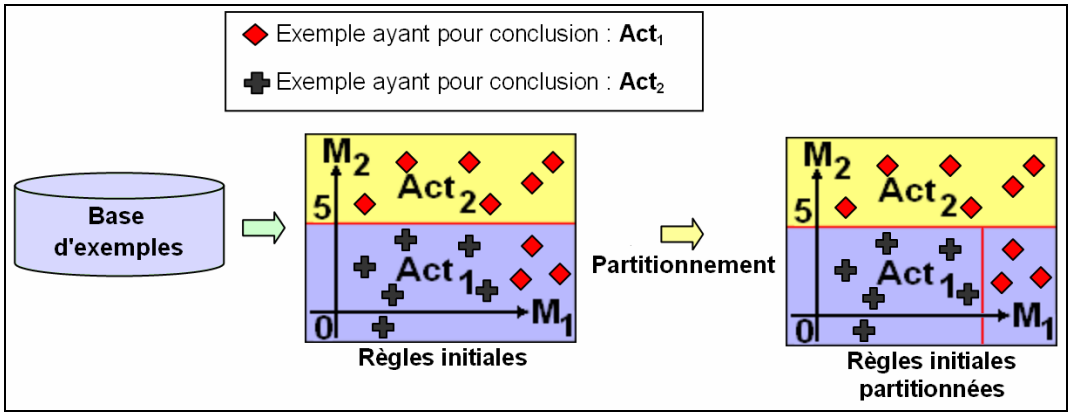


Figure D.5 Partitionnement de l'espace des mesures en zones admettant une conclusion homogène

Le problème du partitionnement d'un espace continu à *n* dimensions a fait l'objet de nombreuses recherches dans de nombreux domaines tels que ceux de l'apprentissage par renforcement (Reynolds, 2000), de la classification non hiérarchique (MacQueen, 1967) ou de l'apprentissage supervisé. Nous disposons, dans notre cadre, d'une base d'exemples exprimée dans un formalisme attributs/valeur nous permettant de nous rapprocher de ce dernier domaine. L'objectif est, en effet, de passer, par induction, d'exemples particuliers (les exemples) à des lois générales (les partitions). En raison des contraintes posées sur le formalisme de représentation des connaissances (base de règles de $\mathcal{BR}$), seuls les algorithmes permettant de construire un modèle prédictif traduisible sous forme de règles de production peuvent être utilisés pour notre application. Des exemples de tels algorithmes sont C4.5 (Quinlan, 1993), RIPPER (Cohen, 1995), Ripple-Down Rules (Gaines & Compton, 1995), etc.. Malheureusement, il n'existe pas d'algorithme d'apprentissage supervisé permettant de prendre directement en compte des connaissances initiales afin de créer des sous-partitions de ces dernières.

Nous proposons donc deux méthodes permettant de repartitionner un espace déjà partitionné (par les règles initiales dans notre cas). La première se base sur la discrétisation de chacune des mesures indépendamment des autres, et la seconde sur l'intersection du partitionnement initial avec un autre partitionnement obtenu par apprentissage supervisé (figure D.6). Nous revenons sur notre première méthode de partitionnement par discrétisations indépendantes des mesures, en partie D.III.3.3.2, et sur de notre méthode de partitionnement par apprentissage et combinaison de règles, en partie D.III.3.3.3.

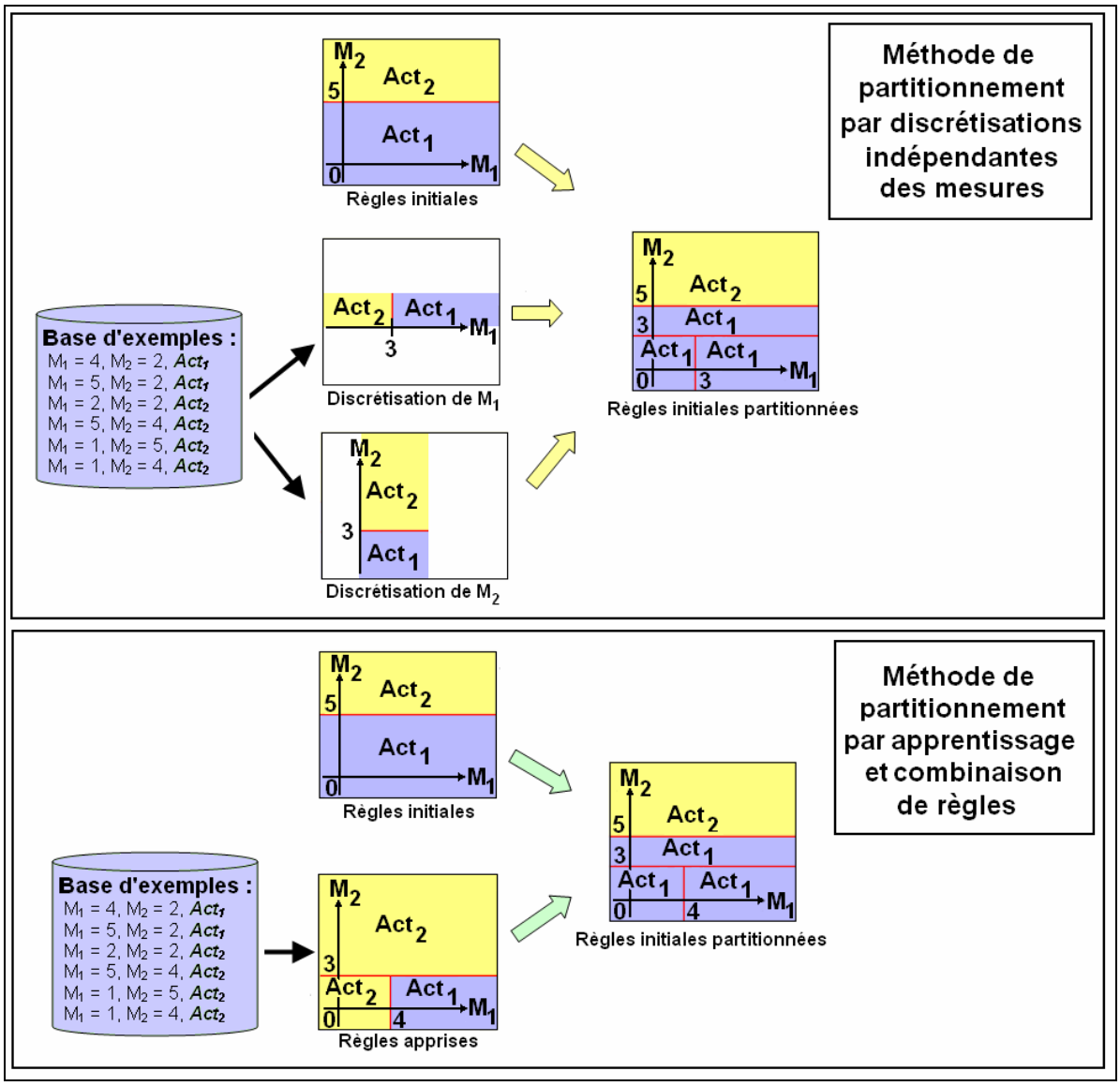


Figure D.6 Méthodes de partitionnement proposées

D.III.3.4.2 Partitionnement par discrétisations indépendantes des mesures

Le principe général de cette méthode est de chercher à discrétiser chaque mesure pertinente en intervalles, puis de reconstruire des prémisses à partir de ces intervalles (en utilisant également les intervalles fournis par la base de règles initiale).

Notre méthode se décompose en 4 étapes (figure D.7) : la première consiste à sélectionner les mesures réellement pertinentes, la seconde à discrétiser ces mesures, la troisième à combiner le jeu d'intervalles ainsi obtenu avec celui provenant des règles initiales et enfin, la dernière, à recomposer des règles à partir des intervalles obtenus.

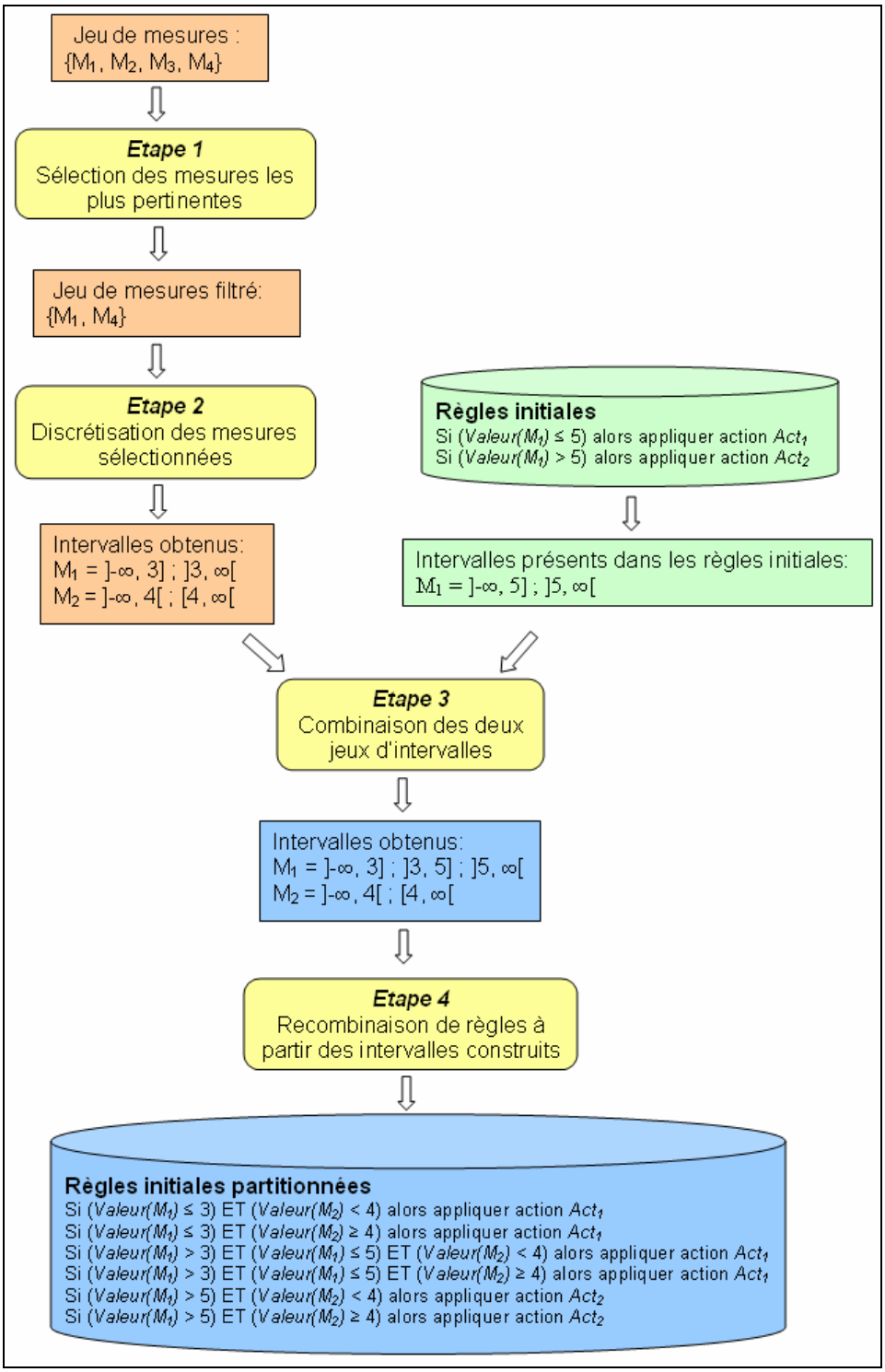


Figure D.7 Méthode de partitionnement par sélection/discrétisation des mesures

Etape 1

Cette première étape consiste à sélectionner les mesures les plus pertinentes. Nous avons déjà évoqué, en partie C.IV.3.2, le problème de la sélection de mesures dans un jeu de mesures. Nous procédons exactement de la même manière dans cette étape, en appliquant un algorithme de sélection d'attributs sur la base d'exemples liée à la base de règles qui nous concerne.

Etape 2
Cette seconde étape consiste à discrétiser les mesures sélectionnées à l'étape 1 en intervalles. De la même manière que pour la sélection de mesures pertinentes, nous avons déjà évoqué ce problème dans le cadre de l'évaluation d'un jeu de mesures (en partie C.IV.3.3). Ainsi, nous proposons également, pour discrétiser ces mesures, de recourir aux techniques de discrétisation supervisée d'attributs continus et d'appliquer ces techniques à notre base d'exemples.

Etape 3
Cette étape consiste, dans un premier temps, à extraire un jeu d'intervalles des règles initiales, puis dans un second temps, à combiner ce dernier avec le jeu d'intervalles construit durant la l'étape 2.

L'extraction du jeu d'intervalles à partir des règles initiales est triviale : en effet, chaque condition de chaque prémisse des règles forme un intervalle.

Ainsi, il sera extrait de la règle :
`Si (Valeur(`M_1`)>5 ET Valeur(`M_1`)<10 ET Valeur(`M_2`)<3) alors appliquer` Act_1
deux intervalles : un pour M_1 :]5,10[et un pour M_2 :]-∞, 3[

Les caractéristiques de nos bases de règles (bases de règles de $\mathcal{BR}$) permettent d'assurer que le jeu d'intervalles tiré de la base de règles initiale couvrira, pour chaque mesure, l'ensemble ρ. Dans le cas d'une mesure associée à la connaissance considérée mais non présente dans la base de règles initiale, l'ensemble d'intervalles extrait est : {]-∞, ∞[}.

Ainsi, un jeu d'intervalles est composé, pour chaque mesure, d'un ensemble d'intervalles couvrant ρ.

Un exemple de jeu d'intervalles est :
$I_a = \{M_1 : \{]-\infty, 5] ;]5, 10[; [10, \infty[\} ; M_2 : \{]-\infty, 3[; [3, \infty[; M_3 : \{]-\infty, \infty[\}\}$

La combinaison des deux jeux d'intervalles est effectuée mesure par mesure.
Soient I_{init} le jeu d'intervalles initial, $I_{discret}$ le jeu d'intervalles construit durant la précédente étape et I_{combi} le jeu d'intervalles résultant de la combinaison de I_{init} et de $I_{discret}$. Nous notons *Intervalles(I, m)*, la fonction retournant l'ensemble d'intervalles correspondant à une mesure *m* dans un jeu *I*.

I_{combi} est composé pour chaque mesure *m* d'un ensemble d'intervalles défini par :

$$Intervalles(I_{combi}, m) = \{ \forall i \in Intervalles(I_{init}, m) \wedge \forall j \in Intervalles(I_{discret}, m), i \cap j\}$$

Par exemple, si notre jeu d'intervalles I_a est combiné avec un jeu I_b défini par :
$I_b = \{M_1 : \{]-\infty, 15] \} ; M_2 : \{]-\infty, \infty[\} ; M_3 : \{]-\infty, \infty[\}\}$

Le jeux d'intervalles I_{ab} résultant de la combinaison de I_a et de I_b sera caractérisé par :
$Intervalles(I_{combi}, M_1) = \{]-\infty, 5] ;]5, 10[; [10, 15];]15, \infty[\}$
$Intervalles(I_{combi}, M_1) = \{]-\infty, 3[; [3, \infty[\}$
$Intervalles(I_{combi}, M_3) = \{]-\infty, \infty[\}$

Etape 4

Cette dernière étape consiste à reconstruire des règles issues du partitionnement à partir des intervalles construits durant l'étape précédente. La conclusion affectée à chaque prémisse est celle proposée par les règles initiales.

Le principe de l'algorithme que nous proposons est le suivant : l'ensemble des mesures du jeu de mesures est parcouru. Pour chaque mesure *m*, l'ensemble des intervalles du jeu d'intervalles est traduit sous forme d'une condition (cf. D.III.3.2.1 pour la liste des conditions possibles). Chacune de ces conditions est ajoutée, par l'intermédiaire d'un connecteur « ET », à chaque prémisse déjà construite.
Une fois l'ensemble des mesures parcouru, chaque prémisse construite est transformée en règle en utilisant, pour conclusion, celle proposée par la base de règles initiale.

```
PartitionneEspace(K_init)
   Initialisation booléen premiereFois ← VRAIE
   Initialisation Ensemble resultatPrem ← {}
   Pour toutes les mesures m du jeu de mesures JM lié à K_init
      Si (premiereFois)
         premiereFois ← FAUX
         Pour tous les intervalles i de Intervalles(I,m) faire :
            Initialisation Condition cond à partir de i
            Ajouter cond à resultatPrem
         Fin pour
      Sinon
         Initialisation Ensemble resultatTemp ← {}
         Pour tous les intervalles i de Intervalles(I,m) faire :
            Pour toutes les prémisses prem de resultatPrem faire :
               Initialisation Condition cond2 à partir de i
               Ajouter la prémisse « prem ET cond2 » à resultatTemp
            Fin pour
         Fin pour
         resultat ← resultat2
      Fin si
   Fin pour

   Initialisation Ensemble resultatRegles = {}
   Pour toutes les prémisses prem de resultatPrem faire :
     Pour toutes les Regles r_i de la base de règles initiale faire :
         Si (prem est une spécialisation de la prémisse de r) faire :
            - Initialisation Regle r avec pour prémisse prem et pour
            conclusion, la conclusion de r_i
            - Ajouter r à resultatRegles
         Fin si
     Fin pour
   Fin pour
   Retourner resultatRegles
```

Nombre maximal de partitions créées

Nous pouvons borner le nombre d'intervalles composant un jeu d'intervalles I_{ab} résultant de la combinaison de deux intervalles I_a et I_b. Nous pouvons en effet démontrer que pour une mesure *m* :

$$card(Intervalles(I_{ab}, m)) \leq card(Intervalles(I_a,m)) + card(Intervalles(I_b,m)) - 1$$

Nous pouvons donc borner le nombre total de partitions construites à partir de deux jeux d'intervalles I_a et I_b par:

$$nb\_partition(I_a, I_b)) \leq \prod_{m \in JM} [card(Intervalles(I_a, m) + card(Intervalles(I_b, m)) - 1]$$

Bilan sur cette première méthode de partitionnement

Nous avons proposé, dans cette partie D.III.3.3.2, une première méthode de partitionnement de l'espace des mesures en sous-espaces. Cette méthode se base sur la discrétisation de chacune des mesures indépendamment des autres, puis sur la reconstruction de règles en prenant en compte les connaissances initiales.

Cette méthode a pour principal avantage d'être simple à mettre en place et facilement contrôlable. Elle est, de plus, évolutive. En effet, elle n'est pas contrainte par un algorithme de sélection d'attributs ou un algorithme de discrétisation de mesures particulier ; ces derniers peuvent être librement choisis.

Elle offre néanmoins le désavantage de s'intéresser à chaque mesure indépendamment des autres. Or, les cas d'application où les mesures sont toutes indépendantes les unes des autres sont extrêmement rares. Nous proposons donc une seconde méthode pour le partitionnement de l'espace des mesures à partir des exemples qui permet, elle, de tenir compte de l'ensemble des mesures en même temps.

D.III.3.4.3 Partitionnement par apprentissage et comparaison de règles

Comme annoncé à la fin de la partie D.III.3.3.1, le principe général de cette seconde méthode de partitionnement de l'espace des mesures est d'apprendre de nouvelles règles à partir des exemples par apprentissage supervisé puis de comparer celles-ci aux règles initiales. Cette méthode se décompose en deux étapes illustrées en figure D.8.

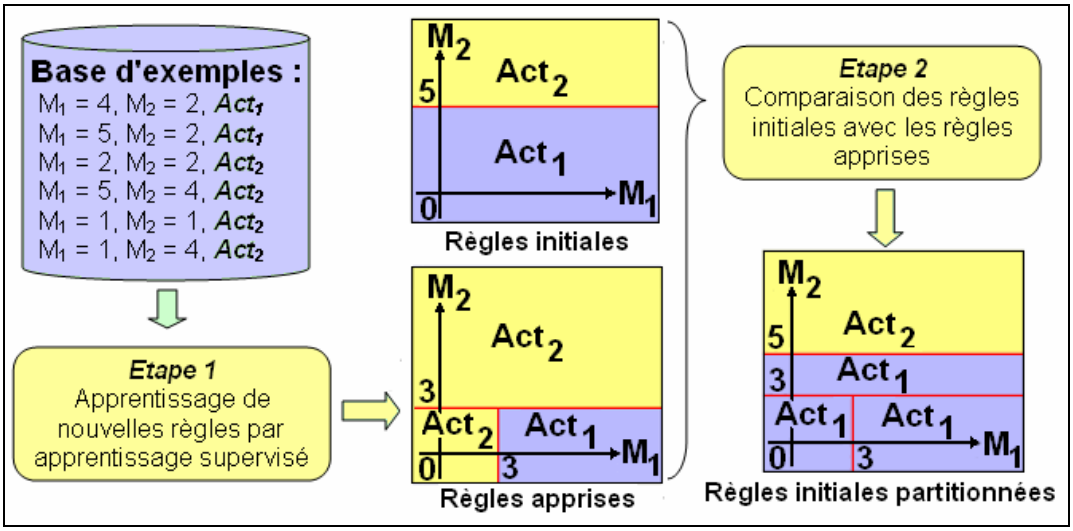


Figure D.8 Méthode de partitionnement par apprentissage et combinaison de règles

Etape 1 : Apprentissage de nouvelles règles

Cette première étape consiste à apprendre, à partir de la base d'exemples (exprimée dans un formalisme attributs/valeur), une base de règles permettant de partitionner l'espace des mesures. Nous utilisons pour cela des techniques d'apprentissage supervisé. L'algorithme d'apprentissage choisi doit être en mesure de fournir un modèle prédictif qui puisse être traduit dans notre formalisme de représentation des connaissances (base de règles de $\mathcal{BR}$). Nous disposons, après cette étape, de deux bases de règles qui définissent chacune un partitionnement de l'espace des mesures : la base de règles initiale et une base de règles apprise.

Etape 2 : Partitionnement de l'espace des mesures par comparaison des règles initiales avec les règles apprises

Cette seconde étape consiste à partitionner l'espace des mesures en comparant les règles initiales avec les règles apprises. L'objectif est d'obtenir un partitionnement pour lequel chaque zone est à la fois une sous-partition d'une zone du partitionnement défini par la base de règles initiale et une sous-partition d'une zone du partitionnement défini par la base de règles apprise.

Nous définissons une première fonction *PartitionneCond($cond_1$,$cond_2$)* qui compare deux conditions portant sur la même mesure, et qui renvoie un ensemble de conditions représentant les zones (sous-partitions) obtenues après partitionnement de la zone définie par la condition $cond_1$ par la zone définie par la condition $cond_2$. Chacune des conditions correspond, pour la mesure concernée, à un intervalle de l'ensemble des réels ρ.

```
PartitionneCond(cond1,cond2)
  Initialisation Ensemble de conditions resultat ← {}
  Ajouter cond1∩cond2 à resultat
  Ajouter à resultat toutes les conditions résultant de cond1 – cond1∩cond2
  Retourner resultat
```

Par exemple, si l'on a deux conditions $cond_a = (M_1 > 5)$ et $cond_b = (M_1 > 3 \text{ ET } M_1 < 10)$, nous aurons :

- $(M_1 > 5) \cap (M_1 > 3 \text{ ET } M_1 < 10) = (M_1 > 5) \text{ ET } (M_1 < 10)$
- $(M_1 > 5) - [(M_1 > 5) \cap (M_1 > 3 \text{ ET } M_1 < 10)] = \{(M_1 \geq 10)\}$

La fonction *PartitionneCond($cond_a$,$cond_b$)* renverra donc comme partitionnement de la zone définie par $cond_a$, l'ensemble des conditions suivant : $\{(M_1 > 5) \text{ ET } (M_1 < 10) ; (M_1 \geq 10)\}$

Nous définissons, à partir de la fonction *PartitionneCond*, une fonction *PartitionneRègle(r_1,r_2)*, qui compare deux règles r_1 et r_2 et renvoie l'ensemble des règles définissant les zones de l'espace obtenues lors du partitionnement de la zone définie par la règle r_1 par la zone définie par la règle r_2.

Le principe de l'algorithme que nous proposons est le suivant : pour chaque mesure *m* utilisée par la règle r_1 ou par la règle r_2, les conditions de r_1 liées à *m* sont partitionnées par les conditions de r_2 liées à *m* à l'aide de la fonction *PartitionneCond*. Nous obtenons ainsi un

ensemble constitué de une à trois conditions que l'on ajoute par l'intermédiaire d'un connecteur « ET » à l'ensemble des prémisses déjà obtenues. La dernière étape consiste à construire de nouvelles règles à partir de l'ensemble des prémisses en utilisant pour conclusion, la conclusion de r_1.

```
PartitionneRègle(r1, r2)
   Initialisation Ensemble de prémisses resultatPrem ← {}
   Initialisation Ensemble de mesures JM ← {mesure utilisée par la prémisse
   de r1 ou celle de r2}
   Initialisation booléen premiereFois ← VRAIE

   Pour toutes les mesures M de JM faire :
      Initialisation Condition cond1 ← sous condition de r1 pour la mesure M
      Initialisation Condition cond2 ← sous condition de r2 pour la mesure M
      Initialisation Ensemble de prémisses resultatTmp ← resultatPrem
      Initialisation resultatPrem ← {}
      Initialisation Ensemble de conditions res ← PartitionneCond(cond1,cond2)
      Pour toutes les conditions cond de res faire :
         Pour toutes les prémisses pTmp de resultatTmp faire :
            Ajouter à resultatPrem, la prémisse « pTmp ET cond »
         Fin pour
         Si (premiereFois)
            Ajouter cond à resultatPrem
         Fin si
      Fin pour
      premiereFois ← FAUX
   Fin pour

   Initialisation Ensemble de règles resultatRegles ← {}
   Pour toutes les prémisses prem de resultatPrem faire :
      Regle r ← Regle avec pour prémisse prem et pour conclusion, la
      conclusion de r1
      Ajouter r à resultatRegles
   Fin pour
   Retourner resultatRegles
```

Par exemple, si l'on a une règle r_a = « *Si* ($M_1 > 5$ *ET* $M_2 < 7$) *alors appliquer* Act_1 » et une règle r_b = « *Si* ($M_1 > 3$) *ET* ($M_1 < 10$) *alors appliquer* Act_2 », nous aurons :

- Pour M_1 : *PartitionneCond*($(M_1 > 5)$, $(M_1 > 3$ *ET* $M_1 < 10)$) = {$(M_1 > 5$ *ET* $M_1 < 10)$; $(M_1 \geq 10)$}
- Pour M_2 : *PartitionneCond*($(M_2 < 7)$,(*VRAIE*)) = {$(M_2 < 7)$}
- $\Rightarrow$ *PartitionneRègle*(r_a,r_b) = {« *Si* ($M_1 \geq 10$ *ET* $M_2 < 7$) *alors appliquer* Act_1 » ; « *Si* ($M_1 > 5$ *ET* $M_1 < 10$ *ET* $M_2 < 7$) *alors appliquer* Act_1 »}

Cette fonction permet de définir une fonction, *PartitionneEspace*(K_{init},K_{appris}), qui compare la base de règles initiale, K_{init}, à une base de règles apprise K_{appris}, et qui renvoie en résultat la base de règles initiale partitionnée. Le principe de l'algorithme que nous proposons est le suivant : chaque règle initiale est partitionnée par l'ensemble des règles apprises. Pour ce faire, chaque règle initiale est partitionnée par une première règle apprise, puis chaque zone obtenue après cette première partition est repartitionnée par une deuxième règle apprise et ainsi de suite.

```
PartitionneEspace(K_init, K_appris)
   Initialisation booléen premiereFois ← VRAIE
   Initialisation Ensemble de règles resultat ← {}
   Ajouter à resultat toutes les règles contenues dans K_init

   Pour toutes les règles r_app de K_appris faire :
      Initialisation Ensemble de règles resultat2 ← {}
      Pour toutes les règles r_result de l'ensemble resultat faire :
         Initialisation Ensemble res ← PartitionneRegle (r_result, r_app)
         Ajouter à resultat2 toutes les règles contenues dans res
      Fin pour
      resultat ← resultat2
   Fin pour
   Retourner resultat
```

Par exemple, si l'on a une base de règles BR_a composée des règles :

- *Si* $(M_1 > 5$ *ET* $M_2 < 7)$ *alors appliquer* Act_1
- *Si* $(M_1 \leq 5$ *ET* $M_2 < 7)$ *alors appliquer* Act_2
- *Si* $(M_2 \geq 7)$ *alors appliquer* Act_1

et une base BR_b composée des règles :

- *Si* $(M_1 > 3$ *ET* $M_1 < 10)$ *alors appliquer* Act_2
- *Si* $(M_1 \leq 3)$ *alors appliquer* Act_1
- *Si* $(M_1 \geq 10)$ *alors appliquer* Act_1

La base résultante de *PartitionneEspace*(BR_a, BR_b) sera composée des règles :

- *Si* $(M_1 \leq 3$ *ET* $M_2 < 7)$ *alors appliquer* Act_2
- *Si* $(M_1 > 3$ *ET* $M_1 \leq 5$ *ET* $M_2 < 7)$ *alors appliquer* Act_2
- *Si* $(M_1 > 5$ *ET* $M_1 < 10$ *ET* $M_2 < 7)$ *alors appliquer* Act_1
- *Si* $(M_1 \geq 10$ *ET* $M_2 < 7)$ *alors appliquer* Act_1
- *Si* $(M_1 > 3$ *ET* $M_1 < 10$ *ET* $M_2 > 7)$ *alors appliquer* Act_1
- *Si* $(M_1 \leq 3$ *ET* $M_2 > 7)$ *alors appliquer* Act_1
- *Si* $(M_1 \geq 10$ *ET* $M_2 > 7)$ *alors appliquer* Act_1

Nombre maximal de partitions créées

De même que pour notre première méthode, nous pouvons borner le nombre maximal de partitions créées.

Soit k une connaissance à laquelle est lié un jeu de mesures JM. Nous définissons deux bases de règles de $\mathcal{BR}$ pour la connaissance k : BR_a et BR_b.

Nous notons *nb_cond(BR, m)*, la fonction renvoyant, pour une base de règles BR, le nombre de conditions différentes relatives à une mesure m.
Le nombre total de partitions construites à partir de deux bases de règles BR_a et BR_b est borné par :

$$nb\_partition(BR_a, BR_b)) \leq \prod_{m \in JM} [nb\_cond(BR_a, m) + nb\_cond(BR_b, m) - 1]$$

Bilan sur cette deuxième méthode de partitionnement
Nous avons proposé, dans cette partie, une méthode permettant de partitionner l'espace des mesures en sous-espaces par apprentissage et comparaison de nouvelles règles.

A l'image de la première méthode de partitionnement que nous avons présentée (cf. D.III.3.3.2), cette méthode offre l'avantage d'être évolutive. En effet, elle n'est pas contrainte par un algorithme d'apprentissage particulier mais peut être utilisée avec n'importe quel algorithme d'apprentissage, à condition que le modèle prédictif construit par celui-ci puisse être traduit dans notre formalisme de représentation des connaissances.

Un avantage de cette méthode, par rapport à la première présentée, est la prise en compte de l'ensemble des mesures en même temps par l'intermédiaire de l'algorithme d'apprentissage. De ce fait, cette méthode permet de mieux gérer les problèmes de dépendances entres mesures qui sont très fréquents dans le cadre d'applications réelles.

Un désavantage est sa complexité algorithmique. En effet, chaque règle initiale est partitionnée par une première règle apprise, puis chaque zone obtenue après cette première partition est repartitionnée par une deuxième règle apprise et ainsi de suite. Le nombre de comparaisons entre règles effectuées peut donc se révéler extrêmement important lorsque le nombre de règles apprises est élevé. Il est donc important de choisir un algorithme d'apprentissage permettant la construction d'un nombre faible de règles. Une méthode pouvant permettre à l'algorithme d'apprentissage de produire moins de règles est de procéder à un prétraitement du jeu de mesures. Il est ainsi possible d'utiliser un algorithme de sélection d'attributs pour éliminer les mesures peu pertinentes.

D.III.3.4.4 Conclusion sur le partitionnement de l'espace des mesures

Nous avons présenté, dans cette partie D.III.3.3.4, deux méthodes de partitionnement de l'espace des mesures en zones admettant une conclusion a priori homogène. Ces deux approches, basées sur l'utilisation des bases d'exemples, permettent d'affiner la partition proposée par les règles initiales.

Nous proposons, en partie F, une expérimentation menée dans le cadre du modèle AGENT, visant à comparer les résultats obtenus avec ces deux approches.

D.III.3.5 Exploration de l'espace des connaissances possibles

D.III.3.5.1 Introduction

Nous nous sommes intéressés dans la partie précédente (D.III.3.3), au problème de la réduction de la taille de l'espace des connaissances et plus particulièrement à celui du partitionnement de l'espace des mesures en zones admettant une conclusion a priori homogène. Une seconde étape de notre approche générale de révision d'un groupe de connaissances consiste à attribuer à chacune des zones, la meilleure conclusion possible.

Pour résoudre ce problème d'affectation de conclusions, nous proposons de recourir à une méthode similaire à celle utilisée par notre approche générique (cf. D.III.2.1) : nous formalisons le problème d'affectation de conclusions sous la forme d'un problème de recherche d'une affectation maximisant la fonction d'évaluation $Eval(K_{init},G_K,S_{Krévisée},P_n)$.

Cette fonction d'évaluation, définie en partie D.III.2.2, estime, pour un système *S,* la qualité d'un jeu de connaissances *K* sur un échantillon P_n d'instances du problème d'optimisation considéré, en tenant compte des modifications apportées aux connaissances initiales. Utiliser cette fonction nécessite, au préalable, de définir une fonction permettant de calculer le taux de modification des connaissances. Nous présentons, en partie D.III.3.5.2, la fonction que nous proposons pour calculer ce taux dans le cas de nos bases de règles de $\mathcal{BR}$.

Il n'est pas possible de choisir pour chaque connaissance, règle par règle, la meilleure conclusion possible sans tenir compte des autres règles de la base. En effet, à l'image des interdépendances existantes entre connaissances, il existe aussi des interdépendances entre règles. La figure D.9 illustre ceci sur l'exemple de la connaissance relative à l'optimalité des états : le choix de la conclusion affectée à la règle r_2 peut jouer dans le choix de la conclusion à affecter à la règle r_3. En effet, si la conclusion *« optimal »* est choisie pour r_2, la meilleure conclusion à affecter à r_3 (celle qui maximise la fonction de performance) est *« non optimal »*. Au contraire, si la conclusion *« non optimal »* est choisie pour r_2, la meilleure conclusion à affecter à r_3 est *« optimal »*.

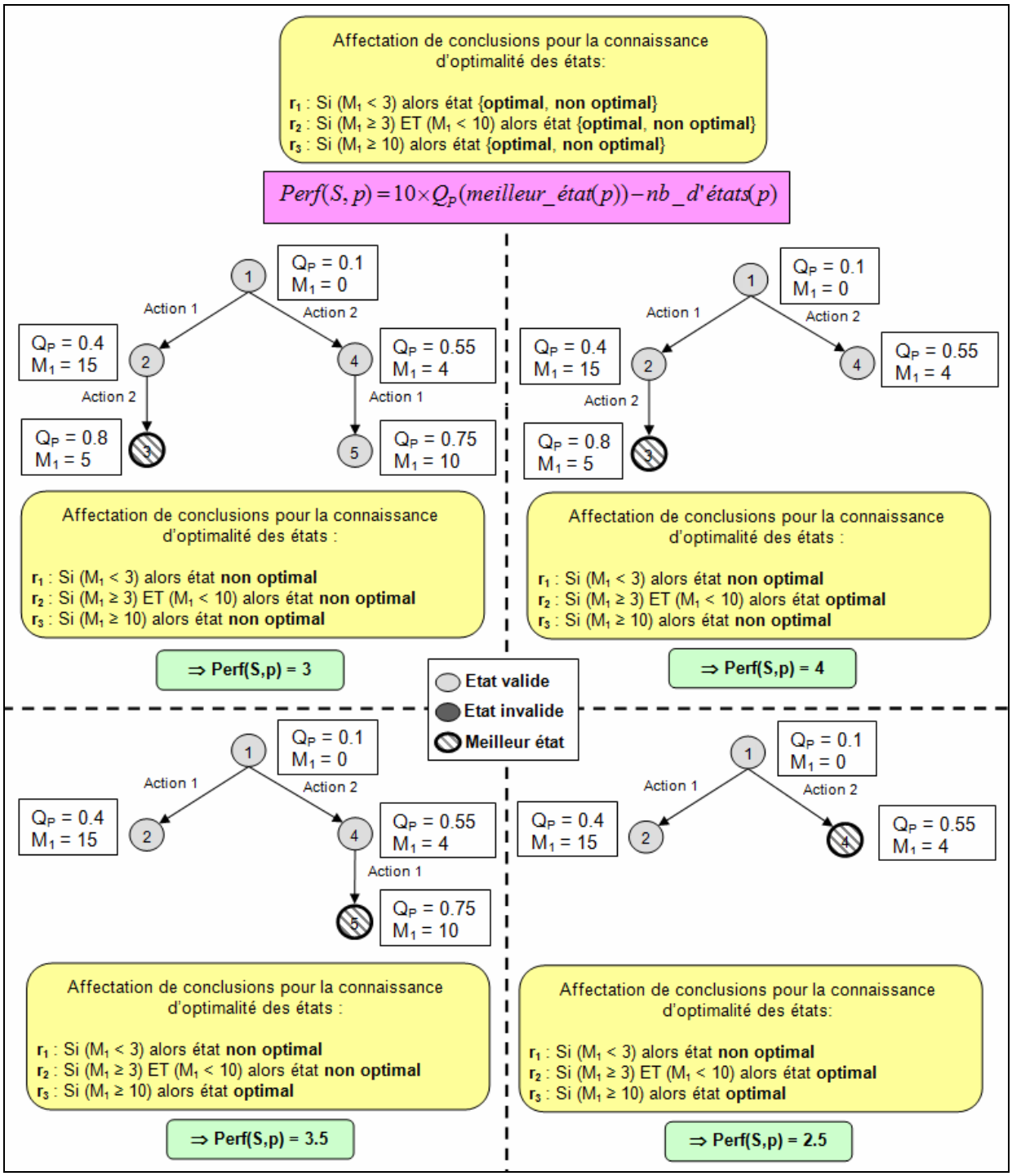


Figure D.9 Problème d'interdépendance entre connaissances

Il est donc nécessaire de tenir compte de toutes les règles en même temps. Il est également nécessaire de tenir compte de l'ensemble des connaissances du groupe simultanément pour éviter les problèmes *d'interdépendance directe* entre connaissances (cf. D.II.2.1.1). Nous rappelons, en effet, qu'un groupe de connaissances est constitué uniquement de connaissances partageant des interdépendances *directes* entre elles. Des connaissances sont en interdépendance *directe* si elles ne peuvent être comprises indépendamment les unes des autres comme c'est le cas des connaissances sur la priorité des contraintes du modèle AGENT (cf. D.II.1).

La taille de l'espace de recherche, pour un groupe de connaissances G_K constitué de *card(G_K)* connaissances, chacune décomposée en un ensemble de partitions {partition}$_k$ et liée à un ensemble de conclusions possibles {conclusion}$_k$, est donc :

$$\prod_{k \in G_K} (card(\{conclusion\}_k))^{card(\{partition\}_k)}$$

La partie D.III.3.5.2 est consacrée à la présentation de la fonction de calcul du taux de modification qui sera utilisée. Nous proposons en parties D.III.3.5.3, D.III.3.5.4 et D.III.3.5.5, différentes approches destinées à répondre à notre problème d'affectation des conclusions.

D.III.3.5.2 Taux de modification pour les bases de règles de $\mathcal{BR}$

Nous avons introduit, en partie D.III.2.2, une fonction d'évaluation *Eval(K_{init},G_K,$S_{Krévisée}$,P_n)* d'un jeu de connaissances. Cette dernière nécessite de définir une fonction *TxModif(k_i,k_j)* qui traduit le taux de modification entre la valeur k_i de la connaissance k et sa valeur k_j. Dans le cadre de notre processus de révision des connaissances, k_i représente la valeur initiale de la connaissance k, et k_j sa valeur révisée.

Nous nous intéressons, dans cette partie D.III.3, à la révision des connaissances représentées sous forme de bases de règles de $\mathcal{BR}$. Une base de règles de $\mathcal{BR}$ a pour caractéristique de former un partitionnement de l'espace des mesures où la prémisse de chaque règle représente une partition de l'espace (cf. D.III.3.3.1). Nous avons proposé dans une première étape de notre processus de révision par analyse de bases de règles de $\mathcal{BR}$ d'affiner, pour chaque base de règles initiale, le partitionnement formé par la base de règles en repartitionnant chacune de ses partitions. La seconde étape de la révision de ces connaissances consiste à attribuer à chaque partition du partitionnement affiné (i.e. à la prémisse d'une règle de $\mathcal{R}$), la meilleure conclusion possible. La fonction *Eval(K_{init},G_K,$S_{Krévisée}$,P_n)* est donc utilisée pour comparer deux affectations possibles de conclusion pour un même partitionnement. k_i et k_j représenteront donc deux bases de règles de $\mathcal{BR}$ constituées des mêmes prémisses : quelles que soient les règles choisies dans k_i, nous pourrons toujours trouver dans k_j, une règle de même prémisse. Seules les conclusions affectées aux prémisses pourront être différentes entre k_i et k_j.

Soient une règle r et une base de règles k. Nous définissons une fonction *RegleMemePremisse(r,k)* qui renvoie les règles de k admettant la même prémisse que celle de la règle r. Nous définissons, de même, une fonction *ConclusionDifferente(r,r')* qui renvoie 1 si la règle r admet une conclusion différente de celle de la règle r' et 0 sinon.

Nous proposons dans notre cadre d'utiliser comme fonction *TxModif(k_i,k_j)*, la fonction suivante :

$$TxModif(k_i, k_j) = \frac{1}{card(k_i)} \sum_{regle_k \in k_i} ConclusionDifférente(regle_k, RegleMemePremisse(regle_K, k_j))$$

Cette fonction donne le taux de conclusions modifiées entre la base de règles k_i et la base de règles k_j. Elle est définie sur l'intervalle [0,1]. Une valeur de 0 signifie que les deux bases de règles sont similaires. Une valeur de 1 signifie que les deux bases de règles ne proposent, pour

aucune règle, jamais la même conclusion. La figure D.10 présente un exemple de résultat obtenu pour la fonction *TxModif* pour deux bases de règles de $\mathcal{BR}$.

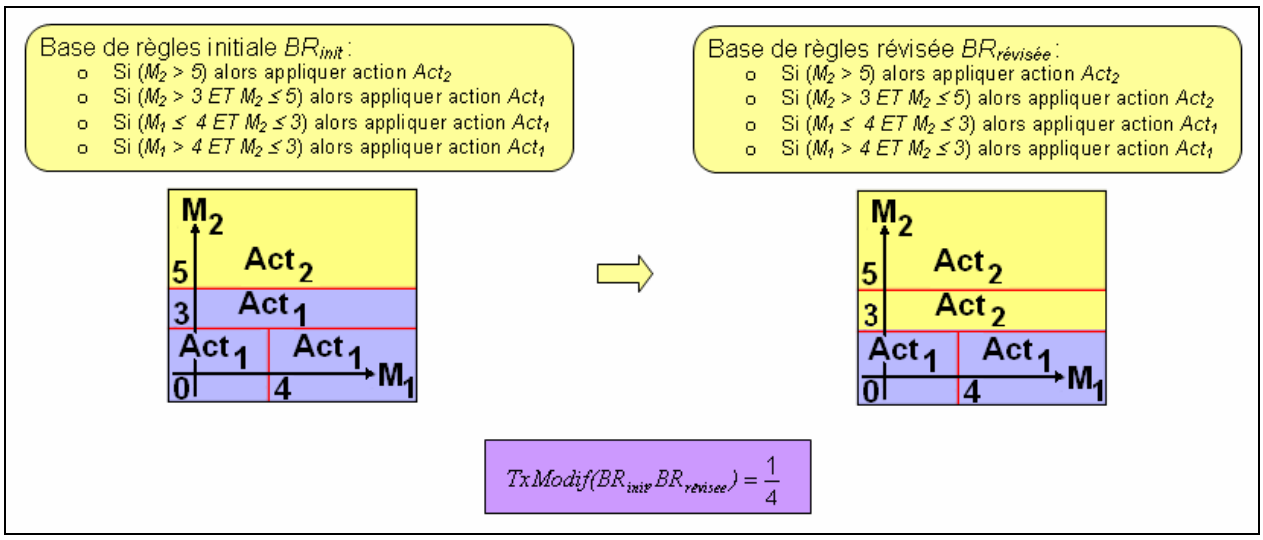


Figure D.10 Exemple de résultat pour la fonction *TxModif*

Il est possible de définir une fonction plus complexe dépendant d'un « poids » affecté à chaque partition. En effet, modifier la conclusion d'une règle très souvent employée peut avoir un impact plus fort que de changer la conclusion d'une règle presque jamais utilisée. La fonction définie peut également prendre en considération d'autres facteurs dépendants du domaine d'application.

D.III.3.5.3 Recherche exhaustive simple

Notre problème d'affectation de conclusion consiste donc à trouver, parmi l'ensemble des affectations possibles, celle qui maximise la fonction d'évaluation $Eval(K_{init}, G_K, S_{Krévisée}, P_n)$.

Une première approche possible pour déterminer ces meilleures affectations est de tester toutes les combinaisons possibles. Ce type d'approche n'est utilisable que pour des cas où la combinatoire est très faible. En effet, même pour un groupe de connaissances composé d'une seule connaissance n'admettant que deux conclusions possibles (comme c'est par exemple le cas pour les connaissances relatives à l'optimalité des états et à la fin de cycle du modèle AGENT), le nombre de combinaisons possibles est de $2^{\text{nb de partitions}}$, ce qui représente plus d'un million de combinaisons pour 20 partitions.

Dans le cas de combinatoires plus importantes, nous proposons de recourir à des méthodes d'exploration incomplète. Nous avons présenté, en partie D.III.2.3, un court état de l'art de ces méthodes. Nous disposons, pour nous aider à résoudre notre problème d'affectation de conclusion, d'une solution initiale (les connaissances initiales) qui sera souvent bonne. Nous proposons donc, comme évoqué en partie D.III.2.3.3, de privilégier les méthodes d'exploration basées sur les trajectoires et en particulier les méthodes de recherche locale.

Une première approche pour faire face à ce problème est d'utiliser une métaheuristique de recherche locale. Nous présentons dans ce cadre, en partie D.III.3.5.4, une mise en œuvre de ces techniques pour notre problème.

Nous proposons également une seconde approche destinée à un type particulier de connaissances. Cette seconde approche, que nous présenterons en partie D.III.3.5.5, est basée sur le paradigme agent. Nous présenterons enfin comment combiner cette approche agent à une métaheuristique de recherche locale.

D.III.3.5.4 Approche par métaheuristique de recherche locale

D.III.3.5.4.1 Principe général

Nous rappelons que notre approche de révision des connaissances représentées sous forme de base de règles de $\mathcal{BR}$ est basée, dans une première étape, sur le partitionnement de l'espace des mesures, et dans une seconde étape, sur l'affectation d'une conclusion à chacune des partitions formées.

L'importance de la combinatoire ne permettra généralement pas de pouvoir tester de manière exhaustive l'ensemble des combinaisons d'affectation possibles. Nous proposons donc dans ce cadre d'utiliser une métaheuristique de recherche locale afin de faire face à ce problème.

Les métaheuristiques de recherche locale sont très utilisées en recherche opérationnelle. Elles se montrent, en effet, très performantes lorsque la taille des espaces de recherche est importante.

Nous avons déjà présenté un état de l'art de ces techniques en partie D.III.3.4.3. Leur principe est de partir d'une solution initiale et de l'améliorer peu à peu par exploration du voisinage.

Ce type de techniques se prête très bien à notre cadre, car nous disposons d'une solution initiale : les connaissances initiales. Nous souhaitons, de plus, apporter le moins de modifications possible au jeu de connaissances initial. Ainsi, il est plus intéressant pour nous de recourir à des métaheuristiques basées sur une trajectoire (recherche locale) plutôt qu'à des métaheuristiques basées sur une population (algorithmes génétiques, optimisation par colonies de fourmis, etc.).

D.III.3.5.4.2 Mise en application pour notre problème d'affectation de conclusion aux partitions

Les méthodes de recherche locale nécessitent de structurer l'espace des solutions en termes de voisinage. Le choix du voisinage a une très grande influence sur la qualité des résultats.

Dans le cadre de nos travaux, nous proposons de définir le voisinage d'une solution comme la modification de la conclusion affectée à une partition pour une *conclusion voisine*. Ainsi à chaque itération du cycle de recherche, seule la conclusion d'une partition sera modifiée.

La définition de la notion de *conclusion voisine* dépend du type de connaissance considéré. Nous proposons pour notre part deux types de voisinages pour une conclusion : l'un pour les connaissances admettant un ensemble de conclusions ordonné (par exemple la connaissance de priorité d'une contrainte du modèle AGENT) et l'autre pour les connaissances admettant un ensemble de conclusions non ordonné (par exemple pour la connaissance de fin de cycle du modèle AGENT : *{continuer, arrêter}*). Nous définissons ces deux types de voisinages de la manière suivante :

Soit un ensemble de conclusions *{conclusion}* non ordonné. Soit $conclusion_i$ une conclusion telle que $conclusion_i \in \{conclusion\}$. Nous proposons de définir le voisinage de $conclusion_i$ par l'ensemble des autres valeurs possibles pour cette conclusion :

$$voisinage(conclusion_i) = \{conclusion\} - \{conclusion_i\}$$

Soit un ensemble de conclusions *{conclusion}* ordonné. Soit $conclusion_i$ une conclusion telle que $conclusion_i \in \{conclusion\}$. Nous proposons de définir le voisinage de $conclusion_i$ par :

$$voisinage_k(conclusion_i) = \{\{conclusion\}_{i-k \leq j} \cup \{conclusion\}_{i+k \geq j}\}$$

$\{conclusion\}_{i-k \leq j}$ est l'ensemble constitué des k conclusions immédiatement supérieures, au sens de la relation d'ordre, à la conclusion $conclusion_i$.
$\{conclusion\}_{i+k \geq j}$ est l'ensemble constitué des k conclusions immédiatement inférieures, au sens de la relation d'ordre, à la conclusion $conclusion_i$.

Par exemple, pour le cas d'une connaissance dont la conclusion est représentée par un entier, le voisinage sera :

$$voisinage_k(conclusion_i) = \{ \forall\ conclusion_j \in \{conclusion\}_{\geq} \ / \ conclusion_i - k < conclusion_j < conclusion_i + k\}$$

Plus k sera élevé, plus le voisinage sera large et donc plus le temps nécessaire à chaque itération sera important mais plus la décision prise à chaque itération de modification de la solution courante aura de chance d'être pertinente.

Nous testerons dans le cadre de nos expérimentations en partie F plusieurs valeurs de k.

D.III.3.5.5 Recherche par approche agents

D.III.3.5.5.1 Hypothèse d'application de l'approche par agents

Nous rappelons que notre approche générale de révision par analyse consiste à diviser les connaissances par groupes (cf. D.II.2.1.1). Chacun de ces groupes peut être composé d'une ou plusieurs connaissances. Les connaissances composant un même groupe partagent des interdépendances *directes*, c'est-à-dire que chaque connaissance ne peut être comprise indépendamment des autres.

L'approche que nous proposons, dans cette partie D.III.3.4.5, est destinée à la révision d'un type particulier de groupes de connaissances. Nous nous intéressons, en effet, aux groupes de connaissances comprenant plusieurs connaissances et dont chaque connaissance relative à un élément (une action, une contrainte, etc.) est composée de règles représentant la *priorité* de l'élément et ayant pour conclusion possible un entier appartenant à un intervalle *[borne_inf,borne_sup]*. Pour un état donné de l'entité considérée, chaque élément calcule sa *priorité* et l'élément ayant la priorité la plus élevée est prioritaire par rapport aux autres.

Les connaissances relatives à la priorité des contraintes et à l'application des actions du modèle AGENT font partie de ce type de groupes de connaissances. Pour un état donné d'un agent géographique, chaque contrainte calcule sa priorité et la contrainte qui a la priorité la plus élevée est prioritaire. De même pour les connaissances relatives à l'application des actions, pour un état donné d'un agent géographique, chaque action calcule son poids et l'action qui a le poids le plus élevé est prioritaire.

D.III.3.5.5.2 Modélisation agent des partitions

Nous rappelons que nous disposons, pour résoudre notre problème d'affectation de conclusion, d'informations globales (la fonction d'évaluation $Eval(K_{init},G_K,S_{Krévisée},P_n)$) ainsi que d'informations locales sur les connaissances (les bases d'exemples). Nous proposons, à travers cette approche agent, de tirer parti de ces informations supplémentaires afin de faciliter la recherche d'une solution optimale.

Nous rappelons qu'une base d'exemples est construite par connaissance du groupe de connaissances ainsi qu'une base d'exemples supplémentaire commune à l'ensemble des connaissances du groupe. Nous n'utiliserons pour notre approche par agents que cette base d'exemples commune à l'ensemble des connaissances.

Il est possible de déterminer, à partir de la base d'exemples commune, pour un groupe de connaissances donné, les nombres de succès et d'échecs connus par chaque couple (partition, conclusion). Nous disposons ainsi d'informations au niveau même des partitions concernant la qualité des conclusions qui leur sont affectées. Les connaissances d'un même groupe de connaissances étant directement interdépendantes, les succès et les échecs connus par chaque connaissance sont également liés aux conclusions proposées par les partitions des autres connaissances. En effet, prenons l'exemple des connaissances de priorité du modèle AGENT : nous ne pouvons pas connaître le nombre de succès et d'échecs qu'un couple (partition, conclusion) d'une contrainte a connu pour un exemple si nous ne connaissons pas, pour ce même exemple, les conclusions données par les partitions des autres contraintes concernées par cet exemple.
La figure D.11 illustre ce problème. Dans cet exemple, deux contraintes C_1 et C_2 ont été définies pour l'agent géographique. Nous nous intéressons aux succès et aux échecs rencontrés par les connaissances relatives à la priorité de ces deux contraintes. Nous rappelons que la connaissance relative à la priorité d'une contrainte est représentée sous la forme d'une base de règles de $\mathcal{BR}$ calculant, en fonction de la valeur de satisfaction de la contrainte, sa priorité (un entier compris entre 1 et 5). Soit E_a l'exemple de la base d'exemples commune définie dans la figure D.11. E_a est composé d'un état $\{Sat_{C1} = 2 ; Sat_{C2} = 2\}$ et d'une conclusion « C_1 *prioritaire* ». Pour l'état de E_a, la connaissance relative à la priorité de C_1, et plus particulièrement la partition définie par la prémisse $(Sat_{C1} < 3.5)$, donne à C_1 une priorité de 5 alors que la connaissance relative à la priorité de C_2, et plus particulièrement la partition définie par la prémisse $(Sat_{C2} < 4.5)$, donne à C_2 une priorité de 1. La priorité de C_1 est plus élevée que celle de C_2, C_1 est donc prioritaire ce qui est en accord avec la conclusion de l'exemple E_a. Les couples $((Sat_{C1} < 5), Priorité\ C_1 = 5)$ et $((Sat_{C2} < 4.5), Priorité\ C_2 = 1)$ connaissent donc un succès pour cet exemple E_a.

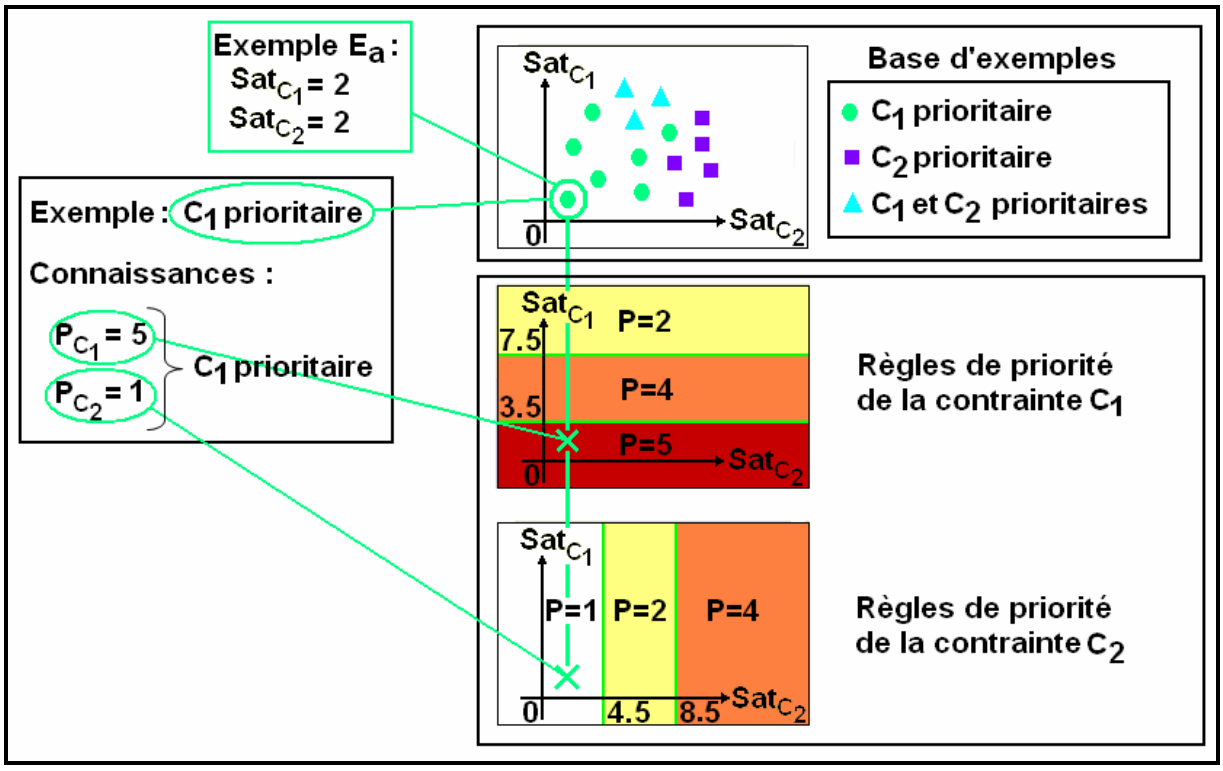


Figure D.11 Analyse des succès/échecs de chaque couple (partition, conclusion) à partir d'une base d'exemples

Cet exemple de calcul des succès/échecs de couples (partition, conclusion) montre qu'il est nécessaire, pour analyser les succès et les échecs rencontrés par des couples (partition, conclusion) dans une base d'exemples, de prendre en considération, pour chaque exemple de la base, l'ensemble des couples concernés par cet exemple.

Afin de profiter des informations obtenues au niveau des partitions, nous proposons de modéliser chaque partition sous la forme d'un *agent partition* chargé de s'affecter la meilleure conclusion possible. Pour cela, un *agent partition* cherche à être le plus satisfait possible, c'est-à-dire à connaître le plus de succès et le moins d'échecs possible. Comme nous l'avons souligné dans le paragraphe précédent, un *agent partition* ne peut prendre de décision seul quant à sa conclusion, il doit, au contraire, s'appuyer sur les conclusions proposées par les autres *agents partition*.

A des fins de clarification, nous formalisons chaque *agent partition* par un triplet (Problème, Connaissance, Prémisse). La prémisse représente la partition (voir partie D.III.3.3.1).

Chaque *agent partition* possède, en plus de ceux utilisés pour le caractériser (le triplet), les attributs suivants :

- Une conclusion qui lui est affectée.
- Une satisfaction courante qui représente la qualité de son état courant. La satisfaction est calculée à partir du nombre de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs.

La figure D.12 présente un exemple d'une partition définie pour la connaissance de priorité de la contrainte C_1 (d'un *agent géographique A*) et modélisée sous la forme d'un *agent partition*.

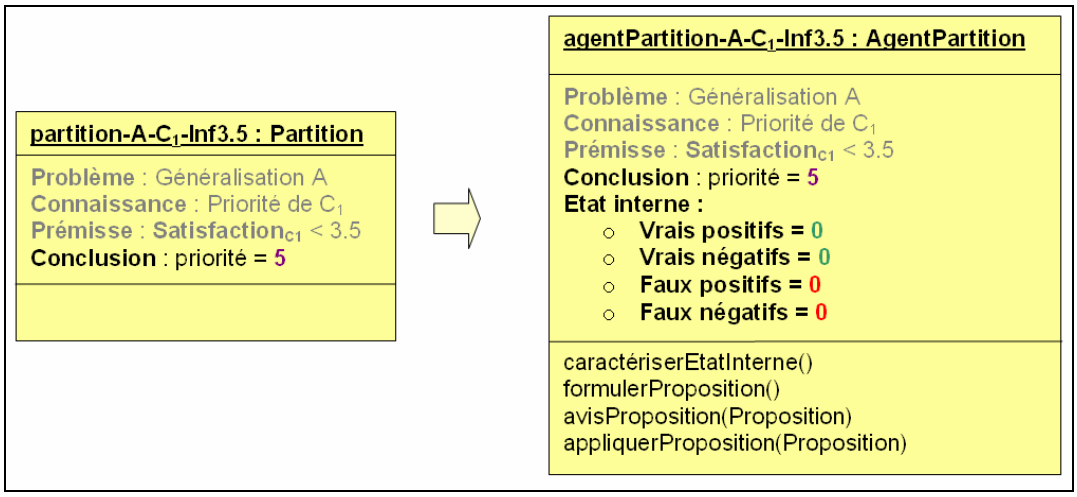


Figure D.12 Exemple de modélisation d'une partition sous forme *d'agent partition*

Un *agent partition* connaît tous les *agents partition* avec qui il partage des exemples et peut envoyer des messages à chacun d'eux. Des *agents partition* partagent un exemple lorsque la prémisse de chacun d'eux est *vraie* pour celui-ci. Par exemple sur la figure D.11, *l'agent partition* (Généralisation de *A*, Priorité de C_1, $(Sat_{C1} < 3.5)$) et *l'agent partition* (Généralisation de *A*, Priorité de C_2, $(Sat_{C2} < 4.5)$) partagent l'exemple E_a.

D.III.3.5.5.3 Processus de recherche des meilleures conclusions

Notre approche agent de recherche des meilleures conclusions se déroule en six étapes (figure D.13).

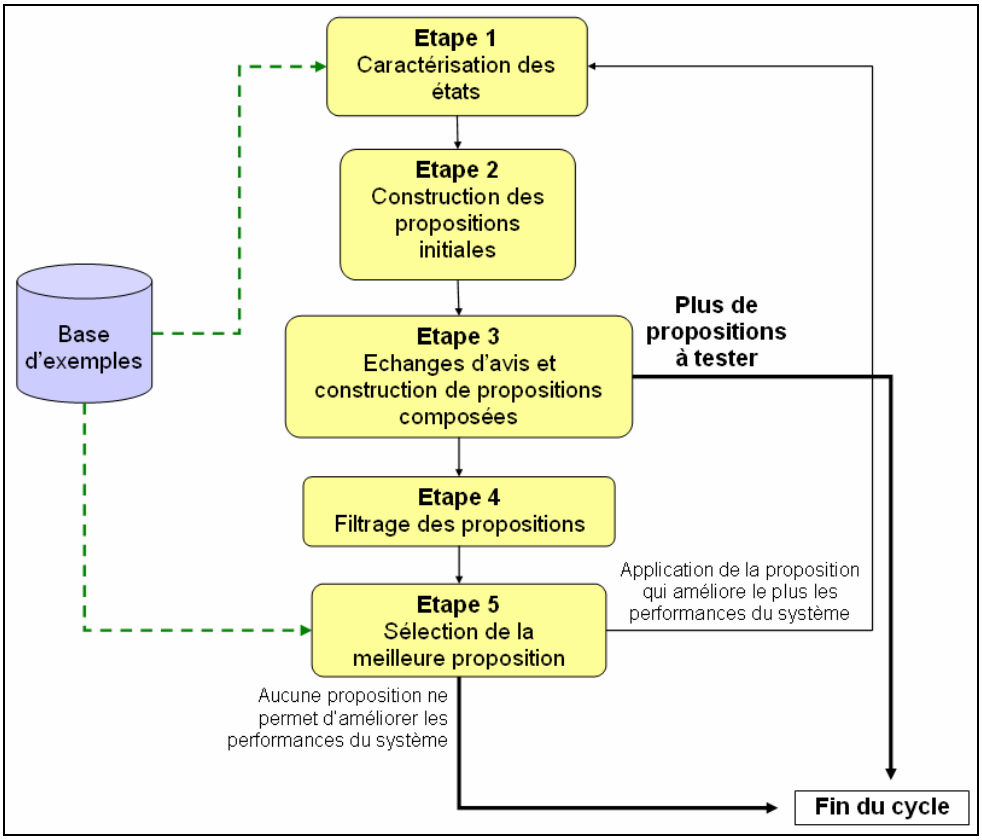


Figure D.13 Approche agent d'affectation des conclusions

Étape 1 : Caractérisation de l'état des *agents partition*

Les *agents partition* déterminent, à partir de la base d'exemples, leur état interne, c'est-à-dire leur nombre de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs. Le calcul de ces statistiques dépend du type de connaissances considéré.

A titre illustratif, nous présentons comment ce calcul est mené pour les connaissances de priorité des contraintes du modèle AGENT. Pour ce type de connaissances, un exemple est formé d'un état caractérisé par les valeurs de satisfaction des contraintes et d'une conclusion qui prend la forme d'un ensemble de contraintes prioritaires pour cet état. Pour un exemple donné, l'*agent partition* partageant cet état qui a la priorité la plus élevée connaît un vrai positif si la contrainte qu'il représente figure parmi l'ensemble des contraintes données comme prioritaires par l'exemple, et un faux positif pour le cas contraire. Les autres *agents partition* partageant l'exemple connaissent un faux négatif si la contrainte qu'ils représentent fait partie de l'ensemble des contraintes prioritaires proposé par l'exemple, et un vrai négatif pour le cas contraire.

La figure D.14 donne un exemple de résultat obtenu pour cette étape de caractérisation. L'exemple E_a de la base concerne les agents *agentPartition-A-*C_1*-Inf3.5* et *agentPartition-A-*C_2*-Inf4.5* et donne la contrainte C_1 comme seule contrainte prioritaire. L'agent *agentPartition-A-*C_1*-Inf3.5* permet à sa contrainte d'être prioritaire (avec une priorité de 5) ; il incrémente donc son nombre de vrais positifs de 1. L'agent *agentPartition-A-*C_2*-Inf4.5* qui a permis à sa contrainte de ne pas être prioritaire (avec une priorité de 1) incrémente son nombre de vrais négatifs de 1.
L'exemple E_b de la base concerne les agents *agentPartition-A-*C_1*-Inf3.5* et *agentPartition-A-*C_2*-Sup8.5* et donne la contrainte C_2 comme seule contrainte prioritaire. L'agent *agentPartition-A-*C_1*-Inf3.5* donne à tort la priorité à sa contrainte (avec une priorité de 5), il incrémente donc son nombre de faux positifs de 1. L'agent *agentPartition-A-*C_2*-Sup8.5* qui n'a pas permis à sa contrainte d'être prioritaire (avec une priorité de 4) incrémente son nombre de faux négatifs de 1.

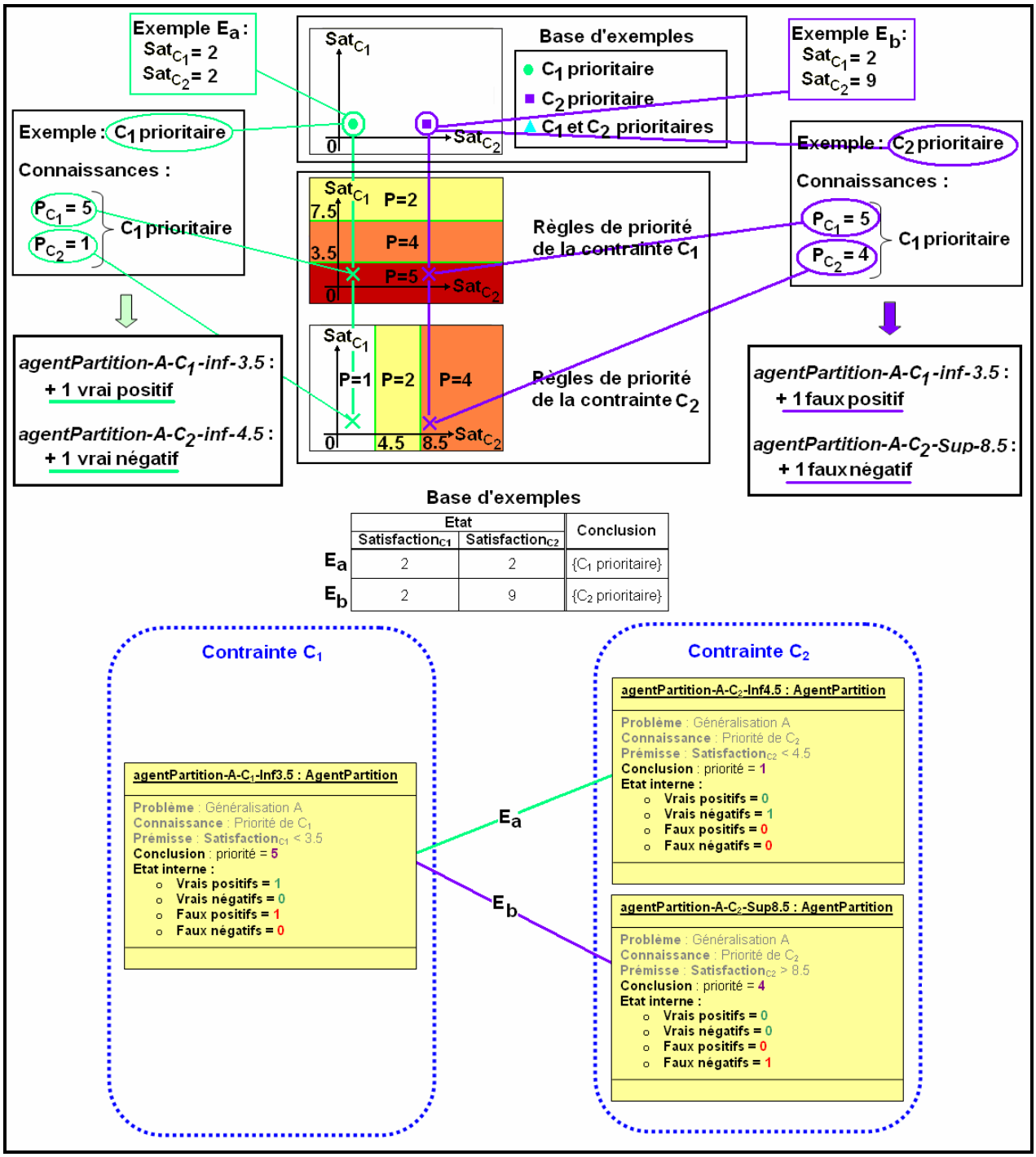


Figure D.14 Exemple de caractérisation

Étape 2 : Construction de propositions de changement de conclusion

Chaque agent partition construit, par analyse de son état interne, une liste de propositions de changement de conclusion. Par exemple, dans le cadre des connaissances de priorité du modèle AGENT, un *agent partition* qui a connu beaucoup de faux positifs (la valeur de priorité de *l'agent partition* a entraîné la fait que la contrainte qu'il représente soit prioritaire alors qu'elle ne devrait pas l'être d'après les exemples issus de l'expérience), proposera de

baisser sa priorité de 2 à 1. Un *agent partition* qui est parfaitement satisfait (qui n'a connu que des vrais positifs et des vrais négatifs sur la base d'exemples) ne construit aucune proposition.

Étape 3 : Avis des *agents partition* sur les propositions et construction de propositions composées

Cette étape consiste à classer les propositions émises, durant la précédente étape, par ordre d'intérêt a priori. Pour établir ce classement, les *agents partition* qui ont formulé des propositions, demandent leur avis à tous les *agents partition* avec lesquels ils partagent des exemples. Cet avis, qui est donné sous la forme d'une *note* et d'une *importance* (qui représente l'importance de l'avis pour cette proposition), permet à *l'agent partition* émettant sa proposition de s'informer sur l'intérêt pour les autres agents de sa proposition. Un *agent partition* établit son avis sur une proposition par analyse de l'évolution présumée que pourrait connaître sa satisfaction si la proposition devenait effective. La *note* est comprise entre 0 et 1 (0 = déconseille fortement l'application de ce mouvement, 1 = conseille fortement l'application de ce mouvement).

Ainsi, un *agent partition* proposant d'augmenter la valeur de sa conclusion (sa priorité) espère diminuer son nombre de faux négatifs. De même un *agent partition* proposant de diminuer la valeur de sa conclusion (sa priorité) espère diminuer son nombre de faux positifs.

Un *agent partition* partageant des exemples avec un *agent partition* proposant d'augmenter la valeur de sa conclusion aura moins de chances a priori d'avoir la plus forte valeur de conclusion. Il espère donc que cette augmentation lui permettra de faire diminuer son nombre de faux positifs mais elle risque en même temps de lui faire diminuer son nombre de vrais positifs. De même, un *agent partition* partageant des exemples avec un *agent partition* proposant de diminuer la valeur de sa conclusion aura plus de chances a priori d'avoir la plus forte valeur de conclusion. Il espère donc que cette diminution lui permettra de faire diminuer son nombre de faux négatifs mais elle risque en même temps de lui faire diminuer son nombre de vrais négatifs.

Soit *ap* un *agent partition*, nous notons *fnegatifs(ap)*, son nombre de faux négatifs, *fpositifs(ap)*, son nombre de faux positifs, *vnegatifs(ap)* son nombre de vrais négatifs et *vpositifs(ap)*, son nombre de vrais positifs. Nous notons également *conclusion(ap)*, la conclusion courante de *ap* et *conclusion(ap,p)* la conclusion de *ap* après application de la proposition *p*. Nous proposons comme note donnée pour chaque agent, les notes et les importances suivantes :

Note donnée par l'*agent partition* ap_{prop} qui formule la proposition p :

- ***A.*** Si $conclusion(ap_{prop}) < conclusion(ap_{prop}, p)$:
  - $Note(ap_{prop}, p) = \frac{fnegatifs(ap_{prop})}{fnegatifs(ap_{prop}) + vnegatifs(ap_{prop})}$
  - $Importance(ap_{prop},p) = fnegatifs(ap_{prop}) + vnegatifs(ap_{prop})$

- ***B.*** Si $conclusion(ap_{prop}) > conclusion(ap_{prop}, p)$:
  - $Note(ap_{prop}, p) = \frac{fpositifs(ap_{prop})}{fpositifs(ap_{prop}) + vpositifs(ap_{prop})}$
  - $Importance(ap_{prop},p) = fpositifs(ap_{prop}) + vpositifs(ap_{prop})$

Note donnée par un *agent partition* ap_{autre} partageant des exemples avec un *agent partition* ap_{prop} qui formule une proposition p :

- ***C.*** Si $conclusion(ap_{autre}) < conclusion(ap_{prop}, p)$ ET $conclusion(ap_{autre}) \geq conclusion(ap_{prop})$:
  - $Note(ap_{autre}, p) = \frac{fpositifs(ap_{autre})}{fpositifs(ap_{autre}) + vpositifs(ap_{autre})}$
  - $Importance(ap_{autre},p) = fpositifs(ap_{autre}) + vpositifs(ap_{autre})$

- ***D.*** Si $conclusion(ap_{autre}) > conclusion(ap_{prop}, p)$ ET $conclusion(ap_{autre}) \leq conclusion(ap_{prop})$:
  - $Note(ap_{autre}, p) = \frac{fnegatifs(ap_{autre})}{fnegatifs(ap_{autre}) + vnegatifs(ap_{autre})}$
  - $Importance(ap_{autre},p) = fnegatifs(ap_{autre}) + vnegatifs(ap_{autre})$

Ainsi, un *agent partition* proposant d'augmenter la valeur de sa conclusion (cas ***A***) espère diminuer son nombre de faux négatifs mais peut en même temps faire diminuer son nombre de vrais négatifs. Plus *l'agent partition* aura connu de faux négatifs et moins il aura connu de vrais négatifs, plus cette proposition sera a priori avantageuse pour lui. L'importance reflète le nombre de succès (les vrais négatifs) et d'échecs (les faux négatifs) en jeu dans cette augmentation de valeur de conclusion.

Similairement au cas ***A***, un *agent partition* proposant de diminuer la valeur de sa conclusion (cas ***B***) espère diminuer son nombre de faux positifs mais peut en même temps faire diminuer son nombre de vrais positifs. Donc plus *l'agent partition* aura connu de faux positifs et moins il aura connu de vrais positifs, plus cette proposition sera a priori avantageuse pour lui. Concernant l'importance, le nombre de succès est ici représenté par le nombre de vrais positifs et le nombre d'échecs par le nombre de faux positifs.

Concernant les *agents partition* qui partagent des exemples avec un *agent partition* qui propose de modifier sa valeur de conclusion, ils ne donneront leur avis que si la proposition émise entraîne des modifications dans leur relation d'ordre par rapport à la valeur de conclusion. En effet, nous nous intéressons, au travers de cette approche par agents, à des connaissances (des base de règles) qui permettent de fournir un classement entre des éléments (par exemples des contraintes). Ce qui importe pour un état donné de l'entité n'est pas tant la valeur de conclusion de chaque élément que le classement des éléments par rapport à leur

valeur de conclusion. Ainsi si la modification de la valeur de conclusion d'un *agent partition* ap_{prop} ne modifie pas son classement au niveau des valeurs de conclusions par rapport à un autre *agent partition* ap_{autre}, le nombre de succès et d'échecs connus par ap_{autre} ne sera pas affecté par cette modification.

Concernant un *agent partition* qui donne son avis sur une proposition d'augmentation de valeur de conclusion émise par un *agent partition* avec lequel il partage des exemples et dont la proposition entrainera une modification de leur relation d'ordre (cas ***C***), celui-ci espère que l'application de la proposition permettra une baisse de son nombre de faux positif mais craint en même temps qu'elle n'entraîne une diminution de son nombre de vrais positifs. L'importance reflète le nombre de succès (les vrais positifs) et d'échecs (les faux positifs) en jeu dans cette augmentation de valeur de conclusion.

Un *agent partition* qui donne son avis sur une proposition de diminution de valeur de conclusion émise par un *agent partition* avec lequel il partage des exemples et dont la proposition entrainera une modification de leur relation d'ordre (cas ***D***), espère que l'application de la proposition permettra une baisse de son nombre de faux négatifs mais craint en même temps qu'elle n'entraîne une diminution de son nombre de vrais négatifs. L'importance reflète le nombre de succès (les vrais négatifs) et d'échecs (les faux négatifs) en jeu dans cette augmentation de valeur de conclusion.

Nous imposons, dans notre approche *agent*, qu'une seule proposition puisse être appliquée par itération du cycle de recherche des meilleures conclusions. Le caractère fini des valeurs prises par les conclusions peut entraîner des effets de bord. Typiquement, si nous disposons de deux *agents partition*, dont l'un avec une conclusion de valeur *borne_sup* et l'autre avec une conclusion de valeur *borne_inf*, il y a de fortes chances pour que la relation d'ordre entre les deux ne puisse jamais être modifiée. Pour limiter ce problème, nous dotons les *agents partition* de la capacité de construire des propositions composées, c'est-à-dire des propositions constituées de deux changements de conclusion de deux agents. Un *agent partition* qui partage des exemples avec un autre *agent partition* formulant une proposition peut ainsi ajouter son propre changement de conclusion. Par exemple, si nous reprenons l'exemple de la figure D.14, si *agentPartition-A-*C_1*-Inf3.5* propose de changer sa priorité de 5 à 4, *agentPartition-A-*C_2*-Sup8.5* peut proposer en plus, dans une même proposition composée, de changer sa priorité de 4 à 5.

Une fois la proposition composée construite, cette dernière est à nouveau proposée aux autres *agents partition* pour qu'ils donnent leurs avis. Les notes et les importances données fonctionnent exactement comme pour une proposition simple sauf qu'elles dépendent cette fois, des deux propositions formulées. Un *agent partition* pourra ainsi se trouver simultanément dans deux cas en même temps. Par exemple un *agent partition* peut formuler une proposition d'augmentation de valeur de conclusion (cas ***A***) et en même temps voir sa relation d'ordre modifiée par rapport à l'autre *agent partition* qui formule, lui, une proposition de diminution de sa valeur de conclusion (cas ***C***).

Ainsi, un *agent partition* formulant une proposition de modification de sa valeur de conclusion (cas ***A*** ou ***B***) et qui voit sa relation d'ordre modifiée par rapport à l'autre *agent partition* formulant une proposition (cas ***C*** ou ***D***) donnera comme note la moyenne des deux notes qu'il aurait dû donner pour chacun de ces cas, pondérée par leurs importances, et comme importance, la somme des deux importances.

Soit un agent partition *ap* se trouvant dans deux cas différents et donnant les notes $Note_a(ap,p)$ avec une importance $Importance_a(ap,p)$ et $Note_b(ap,p)$ avec une importance $Importance_b(ap,p)$. La note et l'importance données sont :

$$Note(ap, p) = \frac{Importance_a(ap, p) \times Note_a(ap, p) + Importance_b(ap, p) \times Note_b(ap, p)}{Importance_a(ap, p) + Importance_b(ap, p)}$$

$$Importance(ap, p) = Importance_a(ap,p) + Importance_b(ap,p)$$

Dans le cas où un *agent partition* partage des exemples avec les deux *agents partition* formulant des propositions à la fois, nous n'utiliserons pas les formules précédentes, mais les formules suivantes :

Note donnée par un *agent partition* ap_{autre} partageant des exemples à la fois avec ap_{prop1} et ap_{prop2} formulant la proposition composée $pc=\{p_1, p_2\}$:

- ***E.*** Si $conclusion(ap_{autre}) < Max(conclusion(ap_{prop1}, p_1), conclusion(ap_{prop2}, p_2))$ ET $conclusion(ap_{autre}) \geq Max(conclusion(ap_{prop1}), conclusion(ap_{prop2}))$:
  - $Note(ap_{autre}, pc) = \frac{fpositifs(ap_{autre})}{fpositifs(ap_{autre}) + vpositifs(ap_{autre})}$
  - $Importance(ap_{autre}, pc) = fpositifs(ap_{autre}) + vpositifs(ap_{autre})$

- ***F.*** Si $conclusion(ap_{autre}) > Max(conclusion(ap_{prop1}, p_1), conclusion(ap_{prop2}, p_2))$ ET $conclusion(ap_{autre}) \leq Max(conclusion(ap_{prop1}), conclusion(ap_{prop2}))$:
  - $Note(ap_{autre}, pc) = \frac{fnegatifs(ap_{autre})}{fnegatifs(ap_{autre}) + vnegatifs(ap_{autre})}$
  - $Importance(ap_{autre}, pc) = fnegatifs(ap_{autre}) + vnegatifs(ap_{autre})$

Ces formules permettent de prendre en considération le fait que deux propositions qui peuvent avoir une influence sur la relation d'ordre vis-à-vis de *l'agent partition* ont été formulées. Ainsi, si deux *agents partition* formulent chacun une proposition, le premier de diminution de sa valeur de conclusion, le second d'augmentation de sa valeur de conclusion, un troisième *agent partition* peut très bien voir sa relation d'ordre modifiée par rapport au premier (et donc avoir, après modification, une valeur de conclusion supérieure à celui-ci), et avoir après application de la proposition une valeur de conclusion inférieure au second et donc ne toujours pas avoir la valeur de conclusion la plus élevée. L'application de la proposition composée ne changera donc rien pour ce troisième *agent partition* et il n'a donc pas à donner son avis.

De façon analogue aux cas ***A*** et ***B***, nous distinguons deux cas : dans le cas ***E,*** *l'agent partition* devient l'agent qui a la plus forte valeur de conclusion après application de la proposition composée et dans le cas ***F***, il perd ce statut.

Étape 4 : Filtrage des propositions en fonction des avis

Pour chaque proposition, l'agent qui a formulé cette proposition (pour le cas d'une proposition composée, l'un des deux est choisi aléatoirement) calcule la qualité de cette proposition. Cette valeur est une moyenne des notes données par les *agents partition*,

pondérée par leurs importances. Cette valeur de qualité est, comme les notes, un réel compris entre 0 et 1. La liste des mouvements possibles est filtrée en fonction de ces valeurs de qualité et en fonction d'un coefficient de filtrage *CF*. Ce coefficient de filtrage est un nombre réel appartenant à [0,1]. Plus grande sera sa valeur, plus le filtrage sera important.

Si *Note_globale* ≤ *CF* la proposition est supprimée de la liste des propositions possibles.

Ce filtrage sert à limiter le nombre de propositions testées. Plus *CF* aura une valeur proche de 0, moins le système testera de propositions a priori mauvaises et donc plus vite il convergera vers une solution finale. Cependant, il peut ainsi passer à coté de bonnes solutions nécessitant l'application d'une proposition a priori moins bonne.

Si la liste des propositions obtenue après filtrage est vide, le cycle se termine

Étape 5 : Sélection de la meilleure proposition

Tous les *agents partition* qui ont formulé des propositions les testent. Ils appliquent pour cela le changement de conclusion qu'ils ont proposé et évaluent les performances du système (grâce à la fonction $Eval(K_{init},G_K,S_{Krévisée},P_n)$) puis reviennent à leur conclusion initiale. Ils comparent ensuite les performances obtenues après et avant ce changement de conclusion.

Si aucune des propositions n'a permis d'améliorer les performances du système, le cycle se termine

Dans le cas contraire, tous les *agents partition* qui ont formulé une ou plusieurs propositions ayant permis d'améliorer les performances du système comparent entre eux les performances obtenues par l'application de leurs propositions. L'*agent* ou les *agents partition* (dans le cas d'une proposition composée) ayant fait la proposition qui a permis d'obtenir les meilleures performances s'appliquent à nouveau les changements de conclusion qu'ils ont proposés.

Itération à l'étape 1

D.III.3.5.5.4 Bilan de notre approche agent d'affectation de conclusion aux partitions

Nous rappelons que notre approche de révision des connaissances représentées sous forme de bases de règles de $\mathcal{BR}$ est basée sur le partitionnement de l'espace des mesures et sur l'affectation d'une conclusion à chacune des partitions formées.

Nous avons présenté, dans le cadre de cette seconde étape, une approche de recherche des meilleures conclusions basée sur le paradigme agent. Le choix des changements de conclusion est guidé par une analyse locale au niveau d'*agents partition* et sur l'échange d'avis entre ces derniers.

Nous pouvons démontrer la convergence de ce processus de recherche. En effet, chaque changement de conclusion validé correspond à une amélioration globale de la qualité des affectations de conclusion. Or comme nous l'avons précisé auparavant, le nombre de combinaisons possibles d'affectations est fini.

Nous avons introduit dans ce processus, un paramètre *CF*, qui permet de limiter de nombre de propositions testées. Plus ce paramètre a une valeur élevée, moins le nombre de propositions testées est important et donc plus le système doit converger vite, mais plus il a de chances de manquer une bonne solution. Le choix d'une valeur de CF doit donc résulter d'un compromis entre l'efficience de notre processus agent et son efficacité.

Notre approche peut être vue comme une recherche locale gloutonne. En effet, tout comme pour une approche locale, nous partons d'une solution initiale (les connaissances initiales) et nous cherchons à améliorer la qualité de cette dernière par modification pas à pas de cette solution. Le cycle de recherche se termine lorsque plus aucune modification de la solution ne permet d'en améliorer la qualité (calculée par la fonction $Eval(K_{init},G_K,S_{Krévisée},P_n)$).

Une différence existant entre une recherche locale gloutonne classique (cf. D.III.2.3.2.1) et notre approche concerne les modifications autorisées à chaque itération du cycle de recherche. Dans le cadre d'une recherche locale gloutonne classique, ces modifications sont choisies dans le voisinage de la solution courante et le choix de celles-ci ne dépend pas, contrairement à notre approche, des caractéristiques locales de la solution courante (voisinage statique). Notre approche permet de construire la liste des modifications possibles en fonction de l'état des *agents partition* (voisinage dynamique).

L'intérêt de cette approche provient de l'utilisation de ce voisinage dynamique. En effet, de part les connaissances introduites au travers de l'analyse locale de la situation (grâce aux *agents partition*), notre approche permet au système de considérer à chaque itération un ensemble de modifications possibles, a priori plus intéressant que celui considéré dans le cadre d'un voisinage statique classique.

Nous proposons, en partie F, une expérimentation de notre approche menée dans le cadre de la révision des connaissances de priorité des contraintes et d'application des actions du modèle AGENT.

D.III.3.5.5.5 Combinaison de l'approche agent et d'une métaheuristique de recherche locale : introduction d'un filtrage heuristique du voisinage

Nous présentons dans cette partie une approche visant à combiner l'approche agent à une métaheuristique de recherche locale.

En effet, il est possible de tirer parti d'informations supplémentaires (les bases d'exemples) afin d'améliorer a priori les performances des métaheuristiques de recherche locale. Nous proposons, en effet, de filtrer le voisinage d'une métaheuristique à l'aide de nos *agents partition*.

Similairement à notre approche agent, ce filtrage est destinée à la révision d'un groupe de connaissances composé d'au moins deux connaissances, proposant chacune un ensemble de valeurs ordonnées pour conclusion. Son principe est d'utiliser les *agents partition* afin de donner des notes aux solutions voisines de la solution courante permettant ainsi de ne pas tester les solutions peu intéressantes à priori et donc de permettre au système de converger plus vite.

Nous utilisons la même modélisation que pour l'approche agent : chaque partition est modélisée sous la forme d'un *agent partition*. Dans le cadre du filtrage, le rôle des *agents*

partition est de s'échanger des avis sur des propositions afin de pouvoir noter la qualité a priori des différentes propositions (appelées « *mouvements* » dans le cadre de la recherche locale).

Le processus d'exploration reste similaire à celui d'une recherche locale classique. La seule différence concerne la phase de définition des mouvements possibles. Cette phase est ici décomposée en 4 étapes (figure D.15).

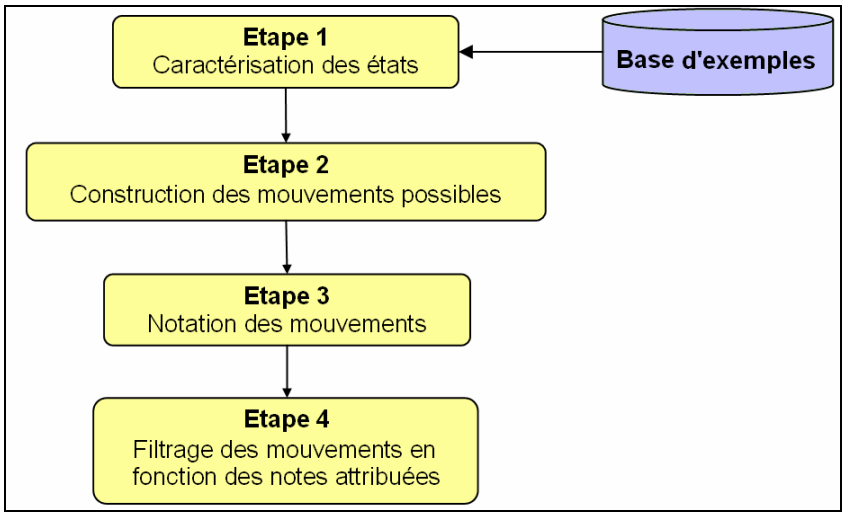


Figure D.15 Approche de filtrage

Étape 1 : Caractérisation de l'état des *agents partition*.

Cette étape est similaire à l'étape 1 de l'approche *agent*. Chaque *agent partition* détermine, à partir de la base d'exemples, son état interne, c'est-à-dire son nombre de vrais positifs, de faux positifs, de vrais négatifs et de faux négatifs.

Étape 2 : Construction des mouvements possibles.

Cette étape est similaire à l'étape de génération des mouvements possibles de l'approche locale classique : les mouvements possibles sont ceux qui permettent de passer de la situation courante à une solution voisine. Nous rappelons que nous avons proposé différents voisinages adaptés à notre problème d'affectation de conclusion en partie D.III.3.5.4.2.

Étape 3 : Echange d'avis et notation des mouvements possibles.

Cette étape est similaire à l'étape 3 de l'approche *agent*, à l'exception que les *agents partition* notent les mouvements et non plus des propositions directement formulées par des *agents partition* et que les *agents partition* n'ont pas la possibilité de formuler des propositions composées. Les *agents partition* notent les mouvements possibles en fonction de leurs états internes et de leurs valeurs courantes de conclusion.

Étape 4 : Filtrage des mouvements en fonction des notes attribuées

Cette étape est similaire à l'étape 4 de *l'approche agent*. Un coefficient de filtrage *CF* est aussi utilisé pour définir le taux de filtrage souhaité.

Bilan du filtrage agent du voisinage

Le filtrage agent du voisinage, à l'image de notre approche agent, est destiné à la révision d'un groupe de connaissances composé d'au moins deux connaissances, proposant chacune pour conclusion un ensemble de valeurs ordonné. Il permet de réduire le nombre de mouvements testés à chaque itération. Dans ce cadre, ce filtrage agent peut permettre de prendre en considération un voisinage plus large (typiquement de prendre une valeur k plus élevée cf. D.III.3.5.4.2).

Nous testerons ce filtrage lors d'expérimentations menées dans le cadre de la révision des connaissances de priorité des contraintes et d'application des actions du modèle AGENT en partie F.

D.III.3.5.6 Bilan de nos approches de recherche des meilleures affectations de conclusion

Notre approche de révision des connaissances représentées sous forme de bases de règles de $\mathcal{BR}$ est basée sur le partitionnement de l'espace des mesures et sur l'affectation d'une conclusion à chacune des partitions formées. Nous avons proposé dans cette partie D.III.3.5, trois approches différentes visant à résoudre le problème de l'affectation des conclusions aux partitions.

La première approche (cf. D.III.3.5.3), qui consiste à tester toutes les combinaisons possibles de conclusions, ne sera utilisable que pour des groupes de connaissances dont l'espace des affectations de conclusions possibles est extrêmement restreint.

Une approche pour faire face à ce problème de taille de la combinatoire est de faire appel aux métaheuristiques de recherche locale. Nous avons présenté dans ce cadre une mise en œuvre de ces techniques pour notre problème en partie D.III.3.5.4.

La troisième approche, que nous avons présentée en partie D.III.3.5.5, est spécialement dédiée au problème de l'affectation de conclusions pour des groupes de connaissances constitués de plusieurs connaissances directement interdépendantes et pour lesquelles l'ensemble des conclusions possibles est ordonné. Cette approche se base sur le paradigme agent et propose de modéliser chaque partition par un *agent partition* qui cherche lui-même à déterminer les meilleurs changements de conclusion possibles. L'un des principaux avantages de cette approche est de permettre de ne tester que très peu de combinaisons possibles d'affectations grâce à une analyse locale préalable, au niveau de chaque partition. Nous avons également proposé dans cette partie d'utiliser notre approche agent afin de filtrer le voisinage utilisé par les métaheuristiques de recherche locale et ainsi de permettre à ces dernières de converger plus rapidement vers une solution.

Nous proposons, au chapitre F, une comparaison de ces différentes approches, menée dans le cadre d'une expérimentation sur la révision des connaissances du modèle AGENT.

D.III.3.6 Simplification des bases de règles

Nous avons proposé, dans les parties précédentes, des approches visant à partitionner l'espace des mesures en zones admettant une conclusion homogène ainsi que des approches visant à attribuer à chaque zone la meilleure conclusion possible.

L'application de ces approches permet d'obtenir des bases de règles révisées. Conformément à la démarche annoncée en partie D.III.3.3, nous nous intéressons maintenant au problème de la simplification des bases de règles obtenues.

L'enjeu de cette simplification est, d'une part, d'améliorer la lisibilité des bases de règles obtenues et, d'autre part, d'améliorer leur performance en limitant les problèmes de sur-apprentissage. Cette simplification répond au principe du rasoir d'Occam qui propose de choisir l'hypothèse la plus simple qui correspond aux données.

L'approche que nous proposons pour simplifier les bases de règles est basée sur les notions de prémisses *compatibles* et *d'agrégation* de prémisses.

L'agrégation de deux prémisses de règles de $\mathcal{R}$ est la prémisse qui correspond à l'union des deux prémisses. Soient deux prémisses p_1 et p_2 de règles de $\mathcal{R}$, l'agrégation p_{12} de p_1 et p_2 est la prémisse de $\mathcal{R}$ telle que $p_{12} = p_1$ OU p_2.
Par exemple, l'agrégation de la prémisse $p_1 = (x < 15)$ ET $(y > 18)$ et de la prémisse $p_2 = (x < 15)$ ET $(y \leq 18)$ est la prémisse $p_{12} = (x < 15)$.

Deux prémisses de règles de $\mathcal{R}$ sont dites *compatibles* si et seulement si l'agrégation de ces dernières est toujours une prémisse pour une règle de $\mathcal{R}$.
Dans l'exemple précédent p_1 et p_2 sont compatibles. Par contre, les prémisses $p_1 = (x < 15)$ ET $(y > 18)$ et $p_3 = (x < 15)$ ET $(y \leq 13)$ ne sont pas compatibles car l'agrégation des deux règles n'appartient pas à $\mathcal{R}$.

Notre approche est également basée sur l'utilisation de notre fonction d'évaluation de la qualité des jeux de connaissances (cf. D.III.2.2). Cette évaluation est calculée sur un échantillon P_k d'instances du problème. Afin d'éviter les problèmes de sur-apprentissage, nous proposons d'avoir recours à une méthode inspirée de celles utilisées pour le post-élagage des arbres de décision (Brelow & Aha, 1997). Nous proposons ainsi de diviser l'échantillon de révision en deux parties : deux tiers des instances sont utilisées pour réviser les connaissances et le dernier tiers est utilisé pour simplifier les bases de règles. A noter que pour éviter les problèmes de non représentativité des instances par rapport à l'ensemble des instances disponibles (voir partie C.II.1), nous proposons de former nos deux groupes en respectant les proportions de chaque famille d'instances créée pendant la phase d'exploration (voir partie C.II.2).

Le principe de notre approche de simplification des règles est le suivant (figure D.16) : le système crée, dans une première étape, une liste d'agrégations possibles, c'est-à-dire d'agrégations de deux règles dont les prémisses sont compatibles. Si cette liste est vide, le processus se termine. Dans le cas contraire, le système choisit dans cette liste une agrégation et l'applique. Si les deux règles concernées par l'agrégation admettent la même conclusion, la règle agrégée prend alors cette dernière pour conclusion et le système revient à la première étape du cycle. A noter que l'agrégation de deux règles admettant la même conclusion

n'entraine aucune modification de la qualité de la base de règles (en termes de performance du système utilisant cette base de règles). Si les deux règles n'admettent pas la même conclusion, le système teste pour la règle agrégée, les deux conclusions proposées par les deux règles et garde celle qui permet au jeu de connaissances de maximiser la fonction d'évaluation sur P_k. Une fois la conclusion choisie, le système compare ses performances avant et après agrégation. Si les performances sont inférieures, le système annule l'agrégation et retire cette dernière de la liste des agrégations possibles avant de retourner à l'étape de choix d'une agrégation. Si les performances sont égales ou meilleures, le système valide l'agrégation et revient à la première étape du processus.

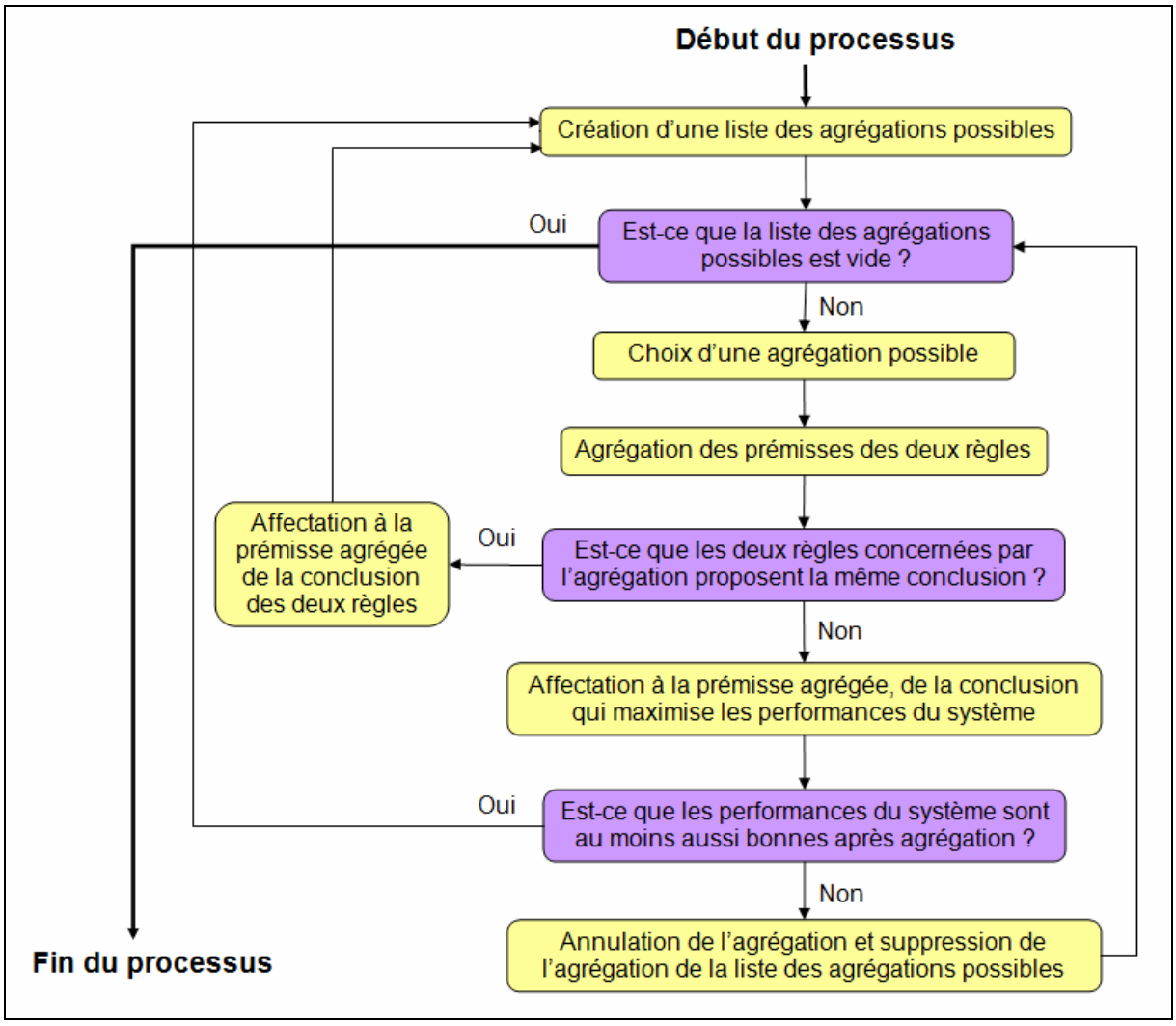


Figure D.16 Approche de simplification des bases de règles

Notre approche demande, dans l'une des étapes, de choisir, parmi une liste d'agrégations possibles, celle qui sera testée en priorité. Ce choix a une influence sur la base de règles obtenues après simplification. La figure D.17 illustre cette influence : on pose dans cet exemple, qu'une règle ayant pour conclusion Act_1 ne peut être agrégée avec une règle ayant pour conclusion Act_2. Dans ce cas, trois agrégations sont possibles à la première itération. En fonction du choix de cette première agrégation, il peut s'avérer impossible de procéder à une seconde agrégation (troisième cas). Nous obtenons dans deux cas la même situation avec 3 règles différentes alors que nous obtenons dans le troisième cas 4 règles différentes.

Le choix de l'agrégation à tester en priorité peut donc entraîner l'obtention de résultats de simplification différents. Ce choix peut aussi influer sur la qualité de la base de règles finale. En effet, dans le cas où l'on permet l'agrégation de règles ayant des valeurs de conclusion différentes, le fait d'obtenir en fin de processus des bases de règles différentes peut entraîner des différences de qualité.

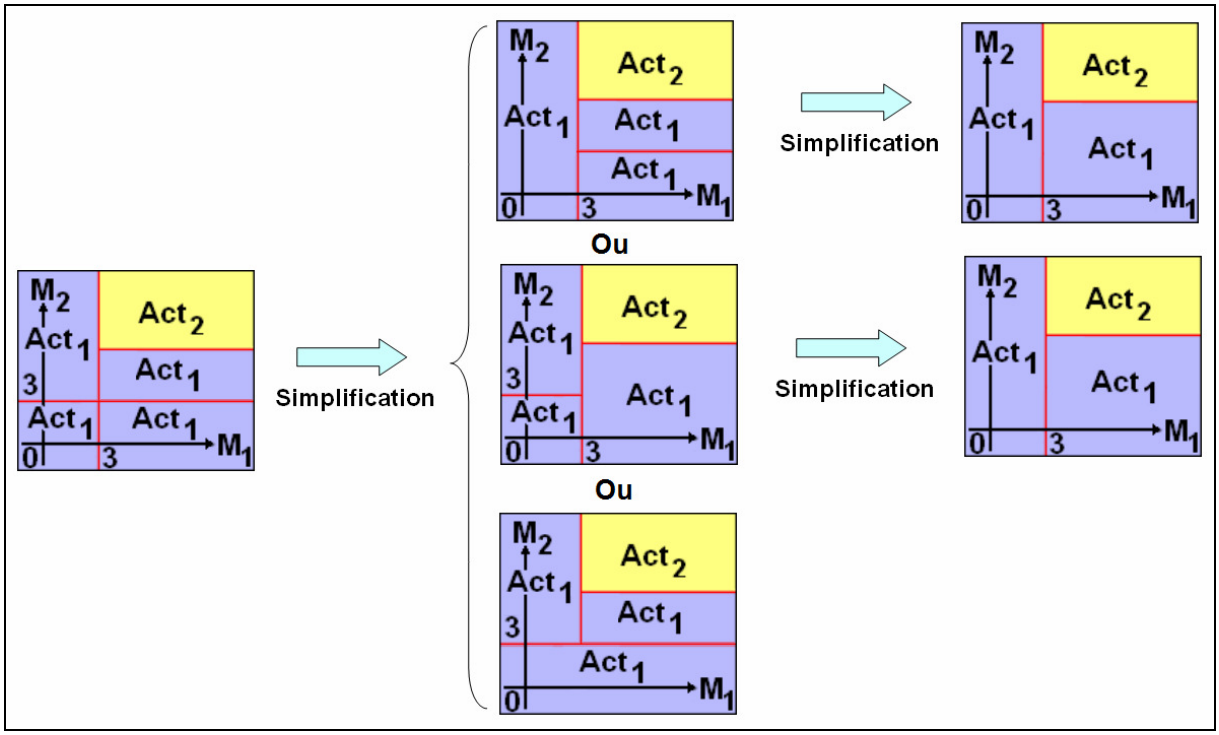


Figure D.17 Influence du choix de l'agrégation sur la base de règles simplifiée

Nous proposons, dans le cadre de ce travail, une heuristique très simple de choix de l'agrégation à appliquer : les agrégations portant sur des règles proposant les mêmes conclusions et n'entraînant pas de modification de la qualité de la base de règles sont appliquées en priorité par rapport aux agrégations portant sur des règles proposant deux conclusions différentes. Cette heuristique permet de privilégier en priorité le regroupement des zones définies comme homogènes par le processus de révision et n'entraînant pas de modification de la qualité de la base de règle.

D.III.3.7 Bilan sur nos approches de révision de bases de règles de $\mathcal{BR}$

Nous avons présenté dans cette partie D.III.3 une approche générale d'analyse des connaissances représentées sous forme d'un type particulier de règles de productions (décrit en partie D.III.3.2.2)

Notre approche s'articule autour de trois grandes étapes : la première consiste à partitionner l'espace des mesures de manière à obtenir un espace fini (cf. D.III.3.4). La seconde étape consiste à attribuer à chaque partition de l'espace, la meilleure conclusion possible (cf. D.III.3.5). La dernière étape consiste enfin à simplifier les bases de règles obtenues (cf. D.III.3.6).

Nous avons proposé, dans le cadre de la partition des jeux de mesures, deux approches visant à repartitionner les partitions déjà définies par les règles initiales en zones admettant une conclusion a priori homogène. Ces approches utilisent les bases d'exemples construites pour former des partitions. La première se base sur la discrétisation de chacune des mesures indépendamment des autres (cf. D.III.3.4.2) et la seconde sur l'intersection du partitionnement initial avec un autre partitionnement obtenu par apprentissage supervisé (cf. D.III.3.4.3).

Nous avons proposé trois approches pour l'étape d'affectation des meilleures conclusions possibles aux différentes partitions. La première consiste en une exploration exhaustive de l'ensemble des affectations possibles et ne peut être appliquée que dans le cadre de groupes de connaissances dont l'espace des affectations de conclusion possibles est extrêmement restreint (cf. D.III.3.5.3). La seconde est basée sur l'utilisation d'une métaheuristique de recherche locale. Ainsi les connaissances initiales forment la solution initiale du problème et le système tente d'améliorer cette solution par modification, partition par partition, des conclusions attribuées. La dernière approche, qui est dédiée à un type particulier de groupes de connaissances, est basée sur le paradigme agent : chaque partition est modélisée sous la forme d'un *agent partition* qui a pour rôle de trouver la meilleure affectation de conclusion possible. Pour ce faire, *l'agent partition* dispose d'informations locales (tirées des bases d'exemples) et peut communiquer avec les autres *agents partition*. Nous avons également proposé de combiner cette approche agent à une métaheuristique de recherche locale afin de permettre un filtrage du voisinage.

La dernière étape de notre processus consiste à simplifier les bases de règles afin d'en améliorer la lisibilité ainsi que la généralité. Notre approche s'inspire des travaux menés sur l'élagage des arbres de décision et propose d'agréger deux à deux des règles, en fonction des performances du système, sur un échantillon d'instances du problème d'optimisation différent de celui utilisé pour la révision.

D.IV Bilan

Nous avons présenté dans ce chapitre notre approche générale de révision des connaissances par analyse des traces.

Notre approche est basée sur la décomposition du problème général de révision de l'ensemble des connaissances en sous-problèmes de révision de groupes de connaissances (figure D.18). Nous avons ainsi proposé dans une première partie (cf. D.II) de diviser l'ensemble des connaissances en groupes de connaissances et de réviser séquentiellement chacun de ces groupes de connaissances.

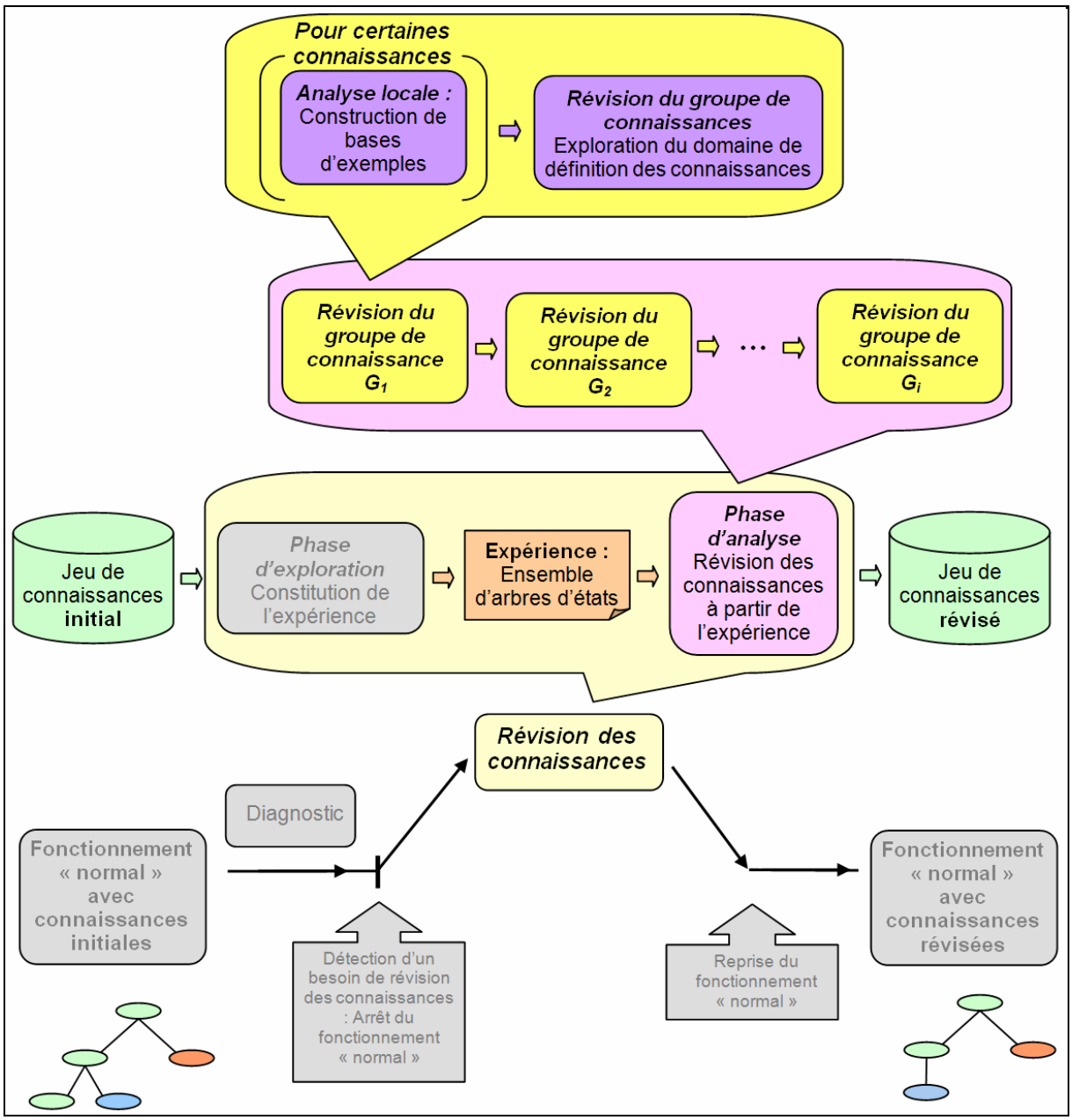


Figure D.18 Approche détaillée de révision des connaissances par analyse des traces

Nous nous sommes intéressés en partie D.III, à la révision des groupes de connaissances. Nous avons pour cela présenté une première approche générique utilisable pour tout type de groupes de connaissances. Cette approche, basée sur l'exploration du domaine de définition des connaissances, est très adaptée aux groupes de connaissances dont le domaine de définition est restreint mais peut se révéler inefficace dans le cas de groupes de connaissances plus complexes. Il sera alors indispensable pour ces groupes de connaissances de proposer des approches de révision plus spécifiques.

Nous avons proposé, dans ce cadre, une approche de révision destinée aux groupes de connaissances représentées sous forme d'un type particulier de bases de règles. Cette approche est basée sur le partitionnement de l'espace des mesures, permettant ainsi de passer d'un problème de définition de règles dans un espace infini à celui de l'affectation de conclusion dans un espace fini. Nous avons ainsi proposé, dans cette partie D.III, deux approches de partitionnement de l'espace des mesures et trois approches de recherche des meilleures affectations de conclusion possibles. Nous avons enfin proposé une approche destinée à simplifier les bases de règles afin de les rendre plus facilement interprétables et d'améliorer leur généralité.

Nous proposons, au chapitre F, des expérimentations, menées dans le cadre du modèle AGENT, visant à tester notre approche générale de révision ainsi que ses différents composants. Nous testons ainsi notre approche d'évaluation des jeux de mesures ainsi que celles destinées à la révision de groupes de connaissances.

Ce chapitre conclut la présentation de notre modèle complet de révision hors ligne des connaissances. Il nous reste maintenant à nous intéresser au problème du déclenchement du processus de révision. Nous présentons, dans le prochain chapitre, un module de diagnostic permettant de définir, lors du fonctionnant « normal » du système, quand les connaissances nécessitent d'être révisées.

Chapitre E
Diagnostic en ligne de la qualité des connaissances procédurales

E.I Introduction

Nous avons présenté, dans les chapitres C et D, notre approche de révision hors ligne des connaissances contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états.

Nous nous intéressons, dans ce chapitre, au problème du déclenchement de ce processus de révision. L'objectif est d'évaluer la qualité des connaissances par analyse en ligne des instances résolues du problème et, ensuite, de déterminer si les connaissances nécessitent ou non d'être révisées (figure E.1). Nous rappelons que nous avons proposé, en partie B.V.2, de recourir à des *agents diagnostic* afin d'assurer ce processus de diagnostic.

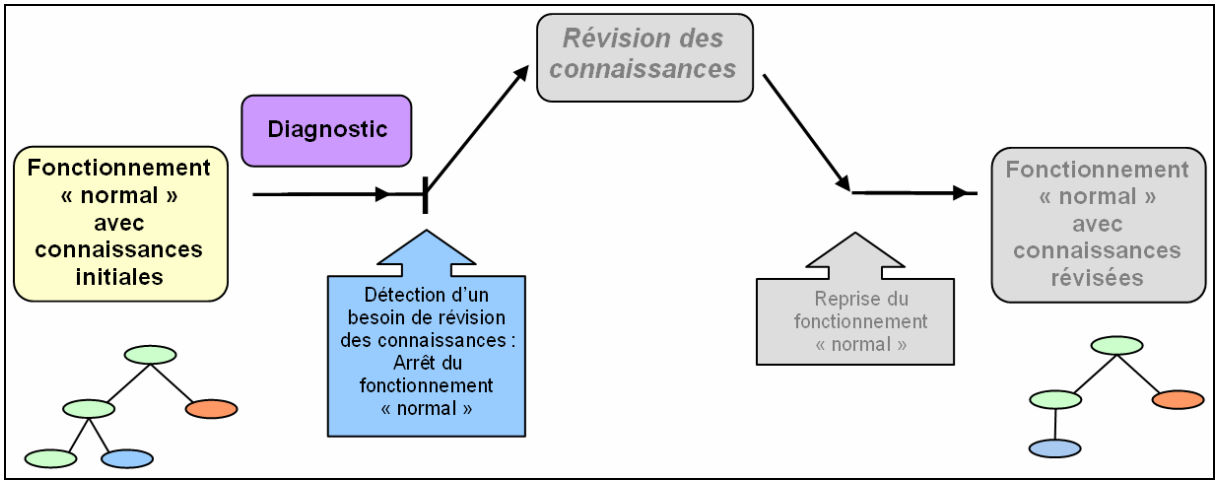


Figure E.1 Diagnostic en ligne de la qualité des connaissances

La partie E.II présente les principales difficultés liées au diagnostic de la qualité des connaissances ainsi que les problématiques qui en résultent.

Nous proposerons ensuite, en partie E.III, une approche de diagnostic en ligne de la qualité des connaissances basée sur l'analyse à la fois locale et globale des instances résolues et sur une analyse multicritère de la qualité générale des connaissances. Nous présenterons, dans ce cadre, deux méthodes d'analyse multicritère qui peuvent être employées pour notre problème de diagnostic: la méthode ELECTRE et la théorie des croyances.

Nous reviendrons enfin, dans une dernière partie (E.IV), sur l'application de notre approche dans le cadre du modèle AGENT.

E.II Problématique du diagnostic en ligne de la qualité des connaissances procédurales

E.II.1 Difficultés du diagnostic en ligne

Nous avons déjà évoqué en partie B.IV.1, les difficultés liées au diagnostic en ligne de la qualité des connaissances.

Nous rappelons que ces difficultés sont de deux types : les interdépendances entre les connaissances et la représentativité des instances traitées (instances sur lesquelles se base le diagnostic).

En effet, nous avons proposé, en partie B.III.2.1, de définir une fonction d'estimation des performances du système de résolution de problèmes sur un échantillon d'instances résolues. Une telle fonction permet d'obtenir des informations globales sur les performances du système mais ne permet en revanche pas de déterminer avec précision quelles connaissances posent a priori problème. Il est donc intéressant de procéder, en plus de cette analyse globale, à une analyse locale au niveau de chaque connaissance. Or, en raison des interdépendances entre connaissances, les analyses locales peuvent s'avérer peu fiables. Ainsi, les succès et les échecs obtenus par une connaissance peuvent dépendre des autres connaissances : typiquement, pour un état donné, l'application d'une action peut entraîner l'exploration de sous-arbres très différents en fonction de l'élagage choisi. Une connaissance pourra donc être considérée à tort comme mauvaise simplement parce qu'une autre connaissance avec qui elle partage des interdépendances est, elle, mauvaise. Ce problème est d'autant plus marqué que le jeu de connaissances utilisé en fonctionnement normal par le système de résolution de problèmes proposera souvent un élagage beaucoup plus sévère que le jeu de connaissances *minimal*. Nous rappelons qu'un jeu de connaissances *minimal* est un jeu de connaissances qui permet de construire l'ensemble des états atteignables par l'ensemble des jeux de connaissances possibles (cf. C.III.1). Ainsi, si l'on peut définir, pour une instance résolue d'un problème, quels états n'auraient pas dû être visités, il n'est en revanche pas possible de savoir si un autre jeu de connaissances aurait permis de trouver un meilleur état. La figure E.2 illustre ce problème : dans cet exemple, les deux connaissances, celle d'application des actions et celle de validité, peuvent prendre respectivement 4 et 2 valeurs différentes. La figure présente les arbres d'états obtenus avec les 8 jeux de connaissances possibles pour une instance d'un problème d'optimisation. Nous considérons que le jeu de connaissances courant est celui ayant entraîné la construction de l'arbre d'états n°7. Ce jeu de connaissances est non *minimal* car il n'a pas permis de visiter les états 6, 7, 8 et 9. Il est possible de déduire, par analyse de cet arbre, que les états 3, 4 et 5 ont été visités inutilement : en effet, visiter les états 1 et 2 aurait suffi à trouver l'état 2 qui est le meilleur état de l'arbre n°7. Une analyse locale au niveau de la connaissance d'application des actions aurait permis de mettre en évidence qu'il était inutile de proposer l'action *Action 2* à l'état 1 et à l'état 2. Par contre, ni cette analyse locale, ni celle qui aurait pu être menée au niveau de la connaissance de validité des états n'auraient permis de savoir qu'il existait des jeux de connaissances permettant de trouver un meilleur état : l'état 7. Les analyses locales d'un arbre d'états apportent des informations sur l'efficience du système de résolution de problèmes (les états inutilement visités) mais pas sur son efficacité (le système atteint-il le meilleur état possible ?).

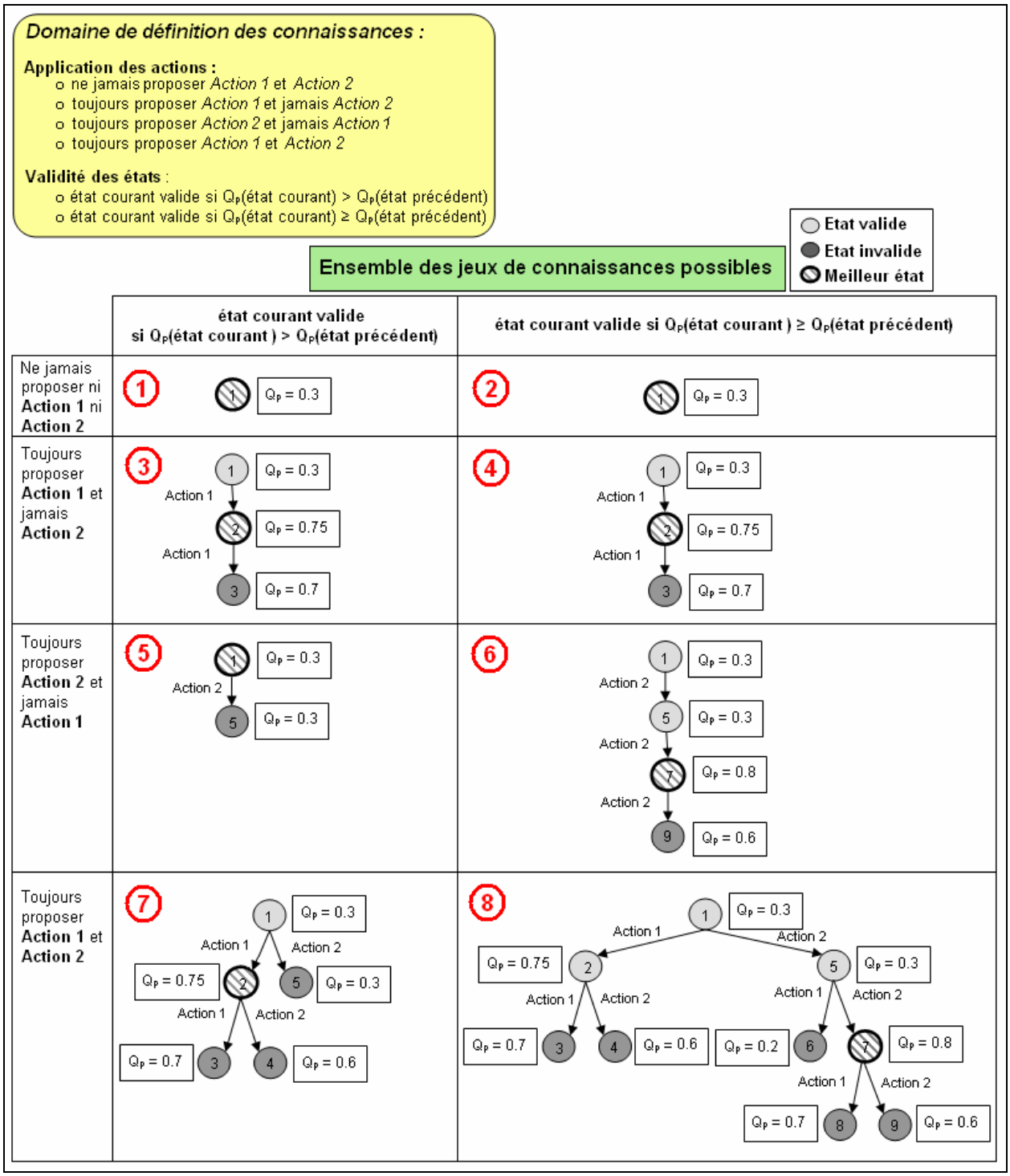


Figure E.2 Problème de l'utilisation d'un jeu de connaissances non *minimal* pour le diagnostic de la qualité des jeux de connaissances

Le second type de difficulté que nous avons évoqué en partie B.IV.1 concerne la représentativité des instances utilisées pour établir le diagnostic. En effet, le diagnostic en ligne se base sur l'analyse d'instances que l'utilisateur a choisi de traiter et non sur des instances sélectionnées judicieusement afin d'assurer la pertinence du diagnostic. Ainsi, il y a un risque que les instances utilisées ne soient que des cas particuliers ne reflétant pas le cas

général. Plus le nombre d'instances utilisées pour le diagnostic est faible, plus ce risque sera élevé.

Une dernière difficulté que nous n'avons pas présentée en partie B.IV.1 concerne le temps pris par le diagnostic de la qualité des connaissances. En effet, si le fait d'introduire un diagnostic en ligne de la qualité des connaissances s'avère handicapant pour les performances du système de résolution de problèmes, l'utilité même de ce diagnostic peut être remise en cause.

E.II.2 Problématique du diagnostic en ligne

L'objectif du diagnostic de la qualité des connaissances est d'évaluer, en fonction des résultats des analyses globales et locales, la qualité du jeu de connaissances utilisé et de fournir un bilan des connaissances qui nécessitent d'être révisées.

Une première question concerne les analyses globales et locales à réaliser.

Une seconde question concerne l'agrégation des résultats de ces analyses afin de fournir une évaluation globale de l'état des connaissances et ainsi déterminer s'il faut ou non proposer à l'utilisateur de déclencher le processus de révision des connaissances.

Enfin une dernière question à se poser, est de savoir quand déclencher le processus de diagnostic pour que celui-ci ne soit pas handicapant pour les performances du système de résolution de problèmes. En effet, plus le nombre d'instances résolues avant que le diagnostic ne soit déclenché est important, plus le module de diagnostic pourra analyser d'instances résolues et donc plus ses conclusions seront fiables. Il ne sera néanmoins pas toujours intéressant de choisir d'attendre qu'un nombre très élevé d'instances soient résolues avant de déclencher le diagnostic. En effet, prenons l'exemple d'un utilisateur devant traiter 1000 instances d'un problème d'optimisation. Il peut être préférable que *l'agent diagnostic* avertisse dès la $50^{ième}$ instance que la qualité du jeu de connaissances est mauvaise afin que l'utilisateur puisse le réviser (automatiquement ou non) plutôt que d'attendre la dernière instance. Il est donc nécessaire de trouver un compromis entre la fiabilité du diagnostic et sa fréquence.

E.III Approche proposée

E.III.1 Principe

Nous rappelons que le module de diagnostic a pour rôle d'évaluer la qualité des connaissances et de déterminer à partir de cette évaluation si une révision des connaissances est nécessaire. Il se base pour cela sur des analyses globales et locales des instances résolues.

Une première difficulté évoquée en partie E.II.1 concerne le risque de diagnostiquer la qualité des connaissances à partir d'instances non représentatives de l'ensemble des instances possibles du problème d'optimisation considéré. Une seconde difficulté mentionnée en partie E.II.1 concerne le risque de perte d'efficience du système de résolution de problèmes dû au diagnostic en ligne de la qualité des connaissances.

Pour faire face à ces deux difficultés à la fois, nous proposons de ne pas diagnostiquer la qualité des connaissances à chaque résolution d'une instance d'un problème d'optimisation, mais de ne diagnostiquer les connaissances qu'une fois qu'un certain nombre d'instances a été résolu. Nous définissons ainsi par *PAS_DIAGNOSTIC* le nombre de résolutions de nouvelles instances nécessaires avant le déclenchement d'un nouveau processus de diagnostic. Comme nous l'avons évoqué en partie E.II.2, le choix de la valeur de *PAS_DIAGNOSTIC* doit résulter d'un compromis entre la fiabilité du diagnostic et sa fréquence. Sa valeur doit être choisie en fonction du système de résolution de problèmes, du type de problème d'optimisation considéré et du nombre d'instances du problème d'optimisation que l'utilisateur doit résoudre.

Notre approche générale de diagnostic est présentée en figure E.3.
Son principe est le suivant : à chaque résolution d'une instance d'un problème, les *agents connaissance* (cf. B.V.2) et *l'agent diagnostic* analysent celle-ci afin de fournir, pour les *agents connaissance*, une analyse locale au niveau de leur connaissance, et pour *l'agent diagnostic*, une analyse globale. Si le nombre d'instances résolues depuis le dernier diagnostic est suffisant (égal à *PAS_DIAGNOSTIC)*, *l'agent diagnostic* déclenche un processus de diagnostic. Ce processus se déroule en deux phases :

- *Phase d'analyse de l'historique* : cette phase consiste, pour chaque agent (*connaissance* et *diagnostic*), à donner une note traduisant la qualité du critère qu'il représente.
- *Phase d'évaluation qualitative du jeu de connaissances* : Les *agents connaissance* évaluent qualitativement leur connaissance en fonction de la note qu'ils lui ont attribuée. *L'agent diagnostic* évalue qualitativement la qualité globale du jeu de connaissances en fonction des valeurs attribuées aux différents critères.

Dans une dernière étape, l'agent diagnostic peut proposer, en fonction du résultat de l'évaluation qualitative des connaissances, de réviser les connaissances.

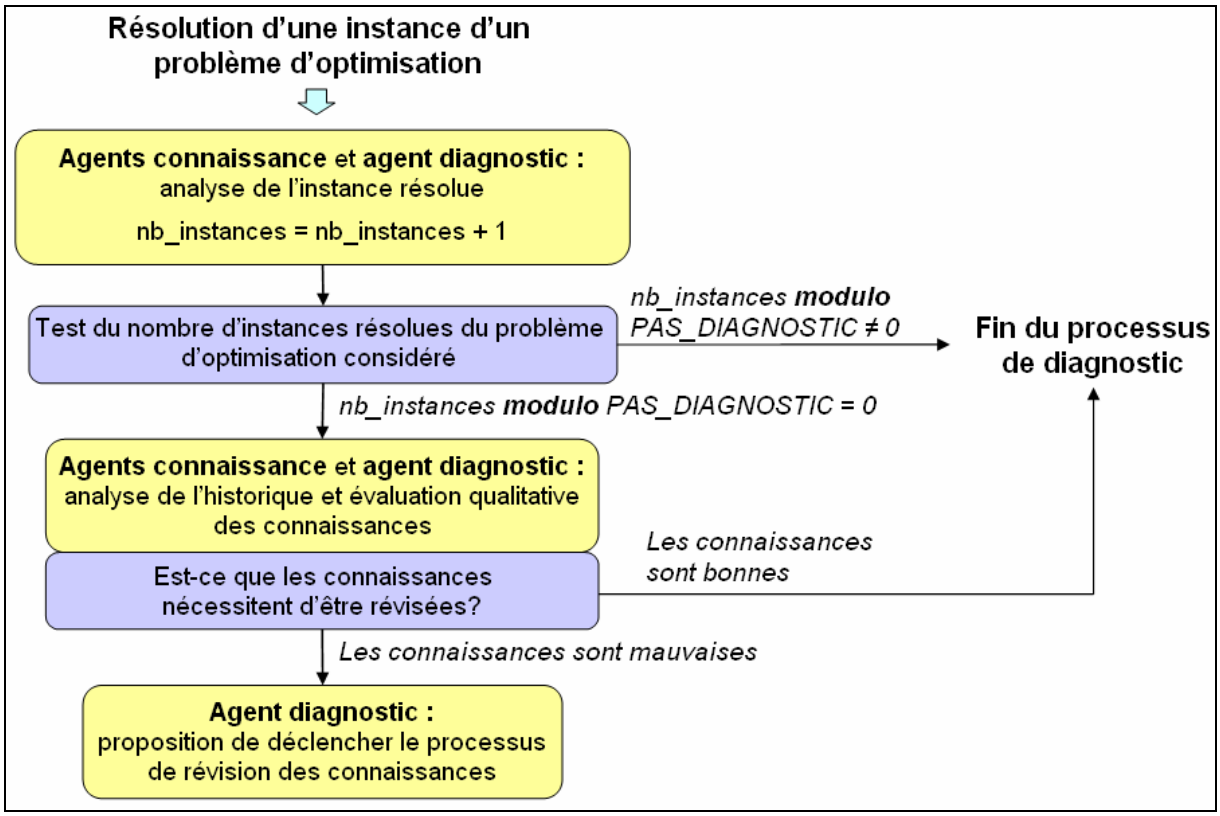


Figure E.3 Approche de diagnostic

Nous détaillons en partie E.III.2, l'étape d'analyse d'une instance résolue d'un problème d'optimisation. La partie E.III.3 est consacrée à la présentation de la phase d'analyse de l'historique et la partie E.III.4 à la phase d'évaluation des connaissances.

E.III.2 Analyse d'une instance résolue d'un problème d'optimisation

A chaque fois qu'une instance d'un problème d'optimisation est résolue, les agents *connaissance* et l'agent *diagnostic* analysent le résultat de cette résolution.

Le rôle des agents *connaissance* est d'analyser localement le résultat de la connaissance qu'ils représentent pour cette instance. Celui de l'agent *diagnostic* est de compléter ces analyses par une analyse globale de l'efficacité du système de résolution de problèmes pour l'instance. En effet, comme nous l'avons montré en partie E.II.1, les analyses locales menées au niveau des connaissances permettent de caractériser l'efficience du système de résolution de problèmes mais ne donne aucune indication sur son efficacité.

Nous proposons de confier la tâche de caractérisation de l'efficacité du système de résolution de problèmes à l'agent *diagnostic* qui a une vision globale des instances résolues. Ainsi, à chaque résolution d'une instance d'un problème, l'agent diagnostic mémorise la qualité Q_P du meilleur état trouvé.

Une autre information intéressante que l'agent *diagnostic* extrait de chaque instance résolue est le taux d'états visités pour arriver au meilleur état (équivalent à la profondeur du meilleur état dans l'arbre) par rapport au nombre d'états total. Ce taux, que nous appelons *taux d'états*

utiles, permet de déterminer si le système de résolution de problèmes a largement développé l'arbre d'états pour arriver au meilleur état ou s'il n'a visité inutilement que très peu d'états.
Nous formalisons ce *taux d'états utiles* de la manière suivante : nous notons *profondeur(p, meilleur_etat(p))*, la fonction qui renvoie la profondeur du meilleur état dans l'arbre d'états construit pour la résolution de l'instance *p* d'un problème d'optimisation. Nous notons *nb_etat(p)*, la fonction qui renvoie le nombre d'états parcourus pour la résolution de l'instance *p* d'un problème d'optimisation. Soit *K*, un jeu de connaissances et *p*, l'instance d'un problème d'optimisation, le *taux d'états utiles (TEU)* est égal à :

$$TEU(K, p) = \frac{profondeur(p, meilleur\_etat(p)))}{nb\_état(p)}$$

Ce taux traduit la fiabilité des analyses locales menées sur l'arbre d'états. En effet, une valeur faible de ce taux indique que le système a beaucoup développé l'arbre d'états et donc que les analyses qui peuvent en être tirées sont a priori faiblement biaisées par l'élagage introduit par les connaissances.

Les agents *connaissance* analysent quant à eux l'arbre d'états obtenu afin de calculer des statistiques sur les connaissances qu'ils représentent. Ces statistiques sont composées des données suivantes :

- *Nombre de faux positifs* : nombre de cas où *l'agent connaissance* a pensé que le concept qu'il représente était vrai alors qu'il ne l'était pas. Par exemple, un *agent connaissance* représentant la connaissance relative à l'optimalité des états qui pense qu'un état donné est optimal alors que l'état ne l'est pas.
- *Nombre de faux négatifs* : nombre de cas où *l'agent connaissance* a pensé que le concept qu'il représente était faux alors qu'il ne l'était pas.
- *Nombre de vrais positifs* : nombre de cas où *l'agent connaissance* a pensé à juste titre que le concept qu'il représente était vrai.
- *Nombre de vrais négatifs* : nombre de cas où *l'agent connaissance* a pensé que le concept qu'il représente était faux.

Notre approche de calcul de ces statistiques est très similaire à notre approche de construction des bases d'exemples (cf. C.IV). Nous proposons, en effet, à l'image de notre approche de construction des bases d'exemples, de construire dans un premier temps la liste des meilleurs chemins puis, de nous baser sur cette liste pour calculer les statistiques souhaitées.

Nous rappelons qu'un meilleur chemin est une séquence d'au moins deux états ayant pour état initial la racine d'un arbre (ou d'un sous-arbre) et pour état final le meilleur état de cet arbre (ou du sous-arbre). L'algorithme de construction de cette liste des meilleurs chemins est donné en partie C.IV.2.3.

Le calcul des statistiques est dépendant du type de connaissances. Des exemples sont donnés dans le cadre du modèle AGENT en partie E.IV.2.

E.III.3 Analyse de l'historique

La phase d'analyse de l'historique intervient lorsqu'un nombre suffisant d'instances du problème considéré a été résolu. L'objectif de cette phase est, pour les agents *connaissance* et *diagnostic*, de fournir une première estimation quantitative de la qualité du critère qu'ils représentent (respectivement la qualité de la connaissance et l'efficacité du système de

résolution de problèmes) à partir des statistiques obtenues après analyse de chaque arbre d'états (cf. E.III.2).

Chaque agent *connaissance* attribue ainsi une note de qualité comprise entre 0 et 1 à sa connaissance. Une note de 0 signifie que la connaissance est, a priori, très mauvaise et une note de 1 signifie que la connaissance est, a priori, parfaite. Le choix de la fonction utilisée pour calculer la note peut dépendre du type de connaissance. En effet, pour certaines connaissances, les erreurs de faux positifs pourront être considérées comme plus problématiques que les erreurs de faux négatifs.

Un indicateur très simple pouvant être utilisé pour le calcul de la note de qualité d'une connaissance est le taux de succès (i.e. le nombre de vrais positifs et de vrais négatifs par rapport à l'ensemble des cas analysés). Nous notons nb_{VP}, le nombre de vrais positifs, nb_{VN}, le nombre de vrais négatifs, nb_{FP}, le nombre de faux positifs et nb_{FN}, le nombre de faux négatifs. La fonction s'exprime de la manière suivante :

$$Note = \frac{nb_{VP} + nb_{VN}}{nb_{VP} + nb_{VN} + nb_{FP} + nb_{FN}}$$

Concernant l'agent *diagnostic*, celui-ci donne une note caractérisant l'efficacité a priori du système de résolution de problèmes. La fonction utilisée pour calculer cette note doit dépendre des préférences de l'utilisateur par rapport au système de résolution de problèmes. Par exemple, certaines applications exigent que le système de résolution de problèmes donne de bons résultats en moyenne, alors qu'il est important pour d'autres que le système soit le moins mauvais possible pour l'ensemble des instances. La définition de cette fonction peut donc nécessiter de disposer de connaissances externes sur le système de résolution de problèmes et en particulier sur la qualité des résultats pouvant être obtenus par celui-ci pour la résolution des instances du problème considéré.

Nous présentons en partie E.IV.3, les fonctions que nous avons définies pour le modèle AGENT.

L'agent *diagnostic* calcule aussi le *taux moyen d'états utiles* pour l'ensemble des instances du problème d'optimisation. Ce taux permet de donner une indication sur la fiabilité de l'ensemble des analyses locales menées. Il est égal à la moyenne des *taux d'états utiles* rencontrés pour chaque instance. Soit K, un jeu de connaissances et P_n, un échantillon d'instances de problème d'optimisation utilisé pour le diagnostic, le *taux moyen d'états utiles (TMEU)* est égal à :

$$TMEU(K, P_n) = \frac{1}{card(P_n)} \sum_{p \in P_n} TEU(K, p)$$

E.III.4 Evaluation qualitative du jeu de connaissances

E.III.4.1 Introduction

L'objectif de la phase d'évaluation des connaissances est d'évaluer qualitativement le jeu de connaissances à partir des notes données durant la phase précédente. Nous rappelons que durant la phase précédente chaque *agent connaissance* et *diagnostic* a donné une note au jeu de connaissances courant. Cette note traduit la qualité du jeu de connaissances par rapport au critère que l'*agent* représente. A des fins de simplification, nous utiliserons, dans cette partie E.III.4, le terme « *critère* » pour désigner un *agent connaissance* ou *diagnostic* et nous utiliserons l'expression « *valeur d'un critère* » pour désigner la note donnée par *l'agent connaissance* ou *diagnostic* représentant ce critère.

Nous définissons pour cela 5 catégories pour qualifier la qualité d'un jeu de connaissances :

- Très mauvaise (TMv)
- Mauvaise (Mv)
- Moyenne (My)
- Bonne (Bn)
- Très bonne (TBn)

L'enjeu de la phase d'évaluation qualitative est de déterminer, à partir du vecteur de notes obtenu par le jeu de connaissances courant, à quelle catégorie de jeu de connaissances celui-ci appartient. Nous proposons pour cela d'exprimer chaque catégorie sous la forme de deux vecteurs de notes représentant, l'un sa borne inférieure, et l'autre sa borne supérieure. Chaque catégorie est donc caractérisée, pour chaque critère, par un intervalle de valeurs possibles pour ce critère. Nous imposons que l'ensemble des intervalles de valeurs associés aux différentes catégories soient disjoints pour chaque critère et qu'ils couvrent l'ensemble des valeurs possibles.

Nous notons $V^{x \to x'}$ le vecteur de notes représentant la frontière entre la catégorie x et la catégorie x'. Soit $V_j^{x \to x'}$ la valeur du critère j pour le vecteur de notes $V^{x \to x'}$ et soit C l'ensemble des critères considérés. Le vecteur de notes $V^{x \to x'}$ peut être décrit par :

$$V^{x} = \left\{ V_j^{x \to x'} \right\}_{j \in C}$$

Dans le cadre de nos cinq catégories de qualité de connaissances, nous définissons les quatre vecteurs de notes *frontières* suivants :

- $V^{TMv \to Mv}$: marque la frontière entre la catégorie « jeu de connaissances de qualité très mauvaise » et la catégorie « jeu de connaissances de qualité mauvaise »
- $V^{Mv \to My}$: marque la frontière entre la catégorie « jeu de connaissances de qualité mauvaise » et la catégorie « jeu de connaissances de qualité moyenne »
- $V^{My \to Bn}$: marque la frontière entre la catégorie « jeu de connaissances de qualité moyenne » et la catégorie « jeu de connaissances de qualité bonne »
- $V^{Bn \to TBn}$: marque la frontière entre la catégorie « jeu de connaissances de qualité bonne » et la catégorie « jeu de connaissances de qualité très bonne »

La figure E.4 présente les différents intervalles de valeurs prises par les différentes catégories pour un critère *j*.

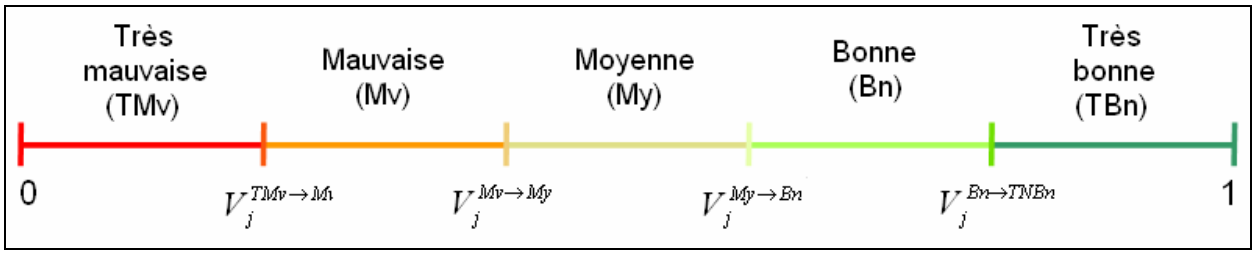


Figure E.4 Catégories pour un critère *j*

L'enjeu de la phase d'évaluation est de déterminer à quelle catégorie le jeu de connaissances appartient par comparaison du vecteur de notes attribué au jeu de connaissances courant (le *vecteur de notes courant*) aux vecteurs de notes frontières. Nous sommes ici dans le cadre d'un problème d'analyse multicritère de type *tri*.

Nous nous intéresserons en partie E.III.4.2 au problème de l'évaluation qualitative du jeu de connaissances courant à l'aide d'une méthode d'analyse multicritère. Nous reviendrons ensuite, en partie E.III.4.3, sur le bilan de qualité pouvant être proposé à l'utilisateur pour l'ensemble des connaissances.

E.III.4.2 Evaluation qualitative du jeu de connaissances

Nous présentons, dans cette partie, un état de l'art des méthodes d'analyse et de décision multicritère. Nous reviendrons ensuite sur les deux méthodes, adaptées de travaux antérieurs, que nous proposons pour résoudre notre problème d'affectation de catégories.

E.III.4.2.1 Etat de l'art des méthodes d'analyse et de décision multicritère

La littérature propose de nombreuses approches d'analyse et de décision multicritère. (Roy, 1985) propose de classer ces approches en trois catégories. La première regroupe toutes les approches qui se basent sur l'agrégation de l'ensemble des critères en un unique critère synthétique. La seconde catégorie regroupe les approches qui permettent d'établir une décision par le biais de comparaisons deux à deux des décisions possibles par analyse de chaque critère. La dernière catégorie regroupe les approches qui se basent sur l'amélioration pas à pas d'une décision initiale.

La première catégorie d'approches est connue sous le terme *d'agrégation complète*. Ces approches consistent à agréger tous les critères en un unique critère par le biais d'une fonction d'utilité. La décision prise est celle maximisant la fonction d'utilité. Un système de poids est souvent utilisé pour prendre en compte l'importance relative des différents critères dans la prise de décision. L'exemple le plus simple et le plus connu de ces méthodes est la moyenne pondérée de notes. Il existe de nombreuses autres méthodes, plus complexes, et fournissant des fonctions d'utilité plus fines (Ignizio, 1978 ; Spronk, 1981 ; Jacquet-Lagreze & Siskos, 1982). Certaines permettent également de prendre en compte des critères flous (Beynon et al., 2000).
Cette catégorie d'approches pose le problème de la commensurabilité (Ben Mena, 2000). Ainsi, chaque critère doit pouvoir être comparé directement aux autres, ce qui implique qu'il soit défini sur une échelle comparable. Une méthode pour résoudre ce problème consiste à normaliser chaque critère (Brauers, 2007). Des travaux portant sur ce type de problématique ont déjà été menés dans le domaine de la généralisation. Ainsi (Ruas & Holzpfel, 2003) se sont intéressés à la normalisation de critères dans le cadre de la caractérisation automatique

des alignements de bâtiments. Une seconde difficulté concerne la définition des poids. Il est en effet souvent complexe d'attribuer à chaque critère un poids réellement conforme aux attentes du décideur. Différents travaux ont porté sur l'aide à la définition des poids (Marques Pereira et al., 2003).
Nous proposons, dans le cadre de ce travail de thèse, de tester une approche de ce type pour notre problème d'évaluation des connaissances. Nous présentons, à cet effet, en partie E.III.4.2.2, une application de la théorie des fonctions de croyance (Shafer, 1976) pour notre problème.

La seconde catégorie d'approches est connue sous le terme *d'agrégation partielle*. Ces approches consistent à comparer deux à deux chaque décision possible. La comparaison est basée sur l'analyse de chaque critère afin d'établir si l'une des deux décisions ne surclasse pas l'autre (Brans & Vinche, 1985 ; Roy, 1991 ; Linden & Stijnen 1995 ; Bana e Costa & Vansnick, 1998).
Ces approches permettent de faire face au problème de l'incomparabilité des différents critères mais fournissent des résultats moins clairs que les approches *d'agrégation complète* (Ben Mena, 2000). En effet, ces approches permettent de comparer deux décisions en comparant chaque critère indépendamment des autres. Contrairement aux approches *d'agrégation complète*, ces approches n'agrègent pas directement les critères entre eux mais uniquement le résultat de leur comparaison.
Nous proposons de tester également une approche de ce type pour notre problème d'évaluation des connaissances. Nous présentons ainsi une application de la méthode ELECTRE TRI (Yu, 1992) pour notre problème en partie E.III.4.2.3.

La dernière catégorie d'approches, dite *d'agrégation locale*, consiste à chercher, à partir d'une solution (décision) initiale, s'il n'y a pas de meilleure solution dans son voisinage (Benayoun et al., 1971 ; Masud & Hwang, 1981 ; Vincke 1989). Ces approches sont généralement basées sur l'échange entre le décideur et le système de prise de décision. Le système de prise de décision construit une solution initiale qui est évaluée par le décideur. Le système modifie alors sa solution en conséquence et la propose de nouveau au décideur qui la réévalue. Ce cycle de « construction d'une nouvelle solution – évaluation » se termine lorsque le décideur est satisfait de la solution proposée. Les nouvelles solutions peuvent être construites automatiquement ou par l'intermédiaire d'experts. Ces méthodes ne sont pas adaptées à notre problématique où nous souhaitons fournir une évaluation des connaissances sans intervention d'experts.

E.III.4.2.2 Evaluation du jeu de connaissances par la méthode des fonctions de croyance

Nous proposons d'appliquer une méthode basée sur la théorie des fonctions de croyance (Shafer, 1976) pour résoudre notre problème de choix de catégorie à affecter à notre jeu de connaissances.

Cette théorie, qui a été appliquée avec succès pour de nombreux problèmes (Le Hégarat-Mascle et al., 2003 ; Omrani et al. 2007 ; Olteanu-Raimond & Mustière, 2008), est particulièrement adaptée à notre problème pour lequel les critères sont peu fiables. En effet, comme évoqué en partie E.II.1, les informations locales servant de base au calcul des notes attribuées aux différentes connaissances seront souvent biaisées en raison des interdépendances existant entre les connaissances. Les notes obtenues pour les connaissances seront donc en général peu fiables.

La théorie des fonctions de croyance permet de gérer :
- l'incertitude des critères
- les connaissances partielles et l'ignorance totale
- les conflits entre critères

Dans le cadre de la théorie des fonctions de croyance, le terme utilisé pour définir les critères est celui de « source », mais, pour des raisons d'uniformisation par rapport aux parties précédentes, nous continuerons à employer le terme « critère ».

Nous proposons, dans le cadre de cette théorie, de ne pas directement comparer le vecteur de notes courant aux vecteurs de notes frontières mais de le comparer à des vecteurs de notes représentatifs calculés à partir des vecteurs de notes frontières. Ces vecteurs de notes représentatifs représentent le « *milieu* » de chaque catégorie.

Nous définissons ainsi un ensemble de cinq vecteurs de notes représentatifs, noté $\{V^{rep}\}$. Soit C, l'ensemble des critères considérés. Un vecteur de notes V^x représentant une catégorie x, dont les jeux de connaissances frontières sont $V^{x'\to x}$ et $V^{x\to x''}$ est caractérisé par :

$$V^x = \left\{ \frac{V_j^{x'\to x} + V_j^{x\to x''}}{2} \right\}_{j\in C}$$

Nous allons chercher, au travers de notre méthode basée sur les fonctions de croyance, à déterminer de quel vecteur de notes représentatif le vecteur de notes courant est le plus proche. Nous pourrons alors déterminer à quelle catégorie le jeu de connaissances appartient.

La théorie des fonctions de croyances exige de définir un univers de référence Θ, appelé cadre de discernement, qui définit l'ensemble des hypothèses pouvant potentiellement répondre au problème considéré. Pour notre application, où l'on cherche quel vecteur de notes représentatif apparier à notre vecteur de notes courant, le cadre de discernement est constitué des cinq vecteurs de notes représentatifs :

$$\Theta = \{V^{TMv}, V^{Mv}, V^{My}, V^{Bn}, V^{TBn}\}$$

Ce cadre de discernement permet de définir un référentiel de définition, noté 2^Θ contenant l'ensemble des combinaisons possibles d'hypothèses:

$$2^\Theta = \{\emptyset, \{V^{TMv}\}, \{V^{Mv}\}, \{V^{M}\}, \{V^{Bn}\}, \{V^{TBn}\}, \{V^{TMv}, V^{Mv}\}, \ldots, \Theta\}$$

Chaque ensemble $\{V^i,\ldots,V^j\}$ est appelé « proposition » et représente le fait que la solution du problème soit l'une ou l'autre des hypothèses de cet ensemble.

La théorie des fonctions de croyance est basée sur l'utilisation de fonctions de croyance. Ces fonctions associent, à une proposition $P \in 2^\Theta$, une masse de croyance, notée $m(P)$, comprise entre 0 et 1. Cette masse de croyance représente le degré de croyance envers cette proposition. Les fonctions de croyance sont définies telles que :

$$\sum_{P\in 2^\Theta} m_j(P) = 1$$

Une question importante dans l'application de la théorie des fonctions de croyance à un problème donné est celle de l'initialisation des masses de croyance. Nous proposons, dans ce cadre, d'utiliser les travaux de (Appriou, 1991) qui proposent de « spécialiser » les critères sur une hypothèse du cadre de discernement (un jeu de connaissances représentatif pour nous) afin que ceux-ci se prononcent uniquement, soit en faveur de l'hypothèse, soit en sa défaveur ou alors ne se prononcent pas par manque de connaissances sur l'hypothèse. Cette proposition permet de définir pour chaque hypothèse, c'est-à-dire chaque vecteur de notes représentatif V^i de $\{V_{rep}\}$, un sous ensemble S_i de 2^{Θ} tel que :

$$S_i = \{V^i, \neg V^i, \Theta, \emptyset\}$$

- V^i : cette proposition signifie que le vecteur de notes courant doit être apparié au vecteur de notes représentatif V^i
- $\neg V^i = \{V^j\}_{V^j \in \{V^{rep}\} \wedge V^j \neq V^i}$: cette proposition signifie que le vecteur de notes courant doit être apparié à un autre vecteur de notes représentatif que V^i
- Θ : cette proposition représente l'ignorance. Le ou les critères ne peuvent se prononcer sur le vecteur de notes à apparier au vecteur de notes courant.
- $\emptyset$: cette proposition représente le conflit entre deux critères ou entre deux hypothèses. Il n'y a jamais de conflit lorsque l'on considère un seul critère et une seule hypothèse.

L'initialisation des masses de croyance revient donc à définir, pour chaque critère, une fonction de croyance fournissant la valeur de la masse de croyance pour les propositions V^i, $\neg V^i$ et Θ. La proposition $\emptyset$ a toujours une masse de croyance égale à 0 à l'initialisation des masses de croyance.

La figure E.5 illustre les fonctions de croyance que nous proposons pour nos différents types de critères. Soit j un critère donné. $V_j^{courant}$ représente la note obtenue par le jeu de connaissances courant pour le critère j et V_j^x représente la valeur de ce même critère j pour le vecteur de notes représentatif V^x.
Nous avons proposé dans cette figure E.5 deux séries de fonctions de croyance différentes : l'une pour le critère d'efficacité et l'autre pour les critères de qualité des connaissances. Concernant le critère d'efficacité, nous souhaitons que de petites variations ne soient pas significatives pour l'appariement de deux jeux de connaissances. Dès qu'un écart important apparaît, le critère doit pouvoir directement rejeter l'appariement. Pour le critère de qualité d'une connaissance, nous souhaitons également que les petits écarts ne soient pas pris en compte pour le rejet d'un appariement mais que les écarts soient pris en compte dès qu'ils dépassent un certain niveau.

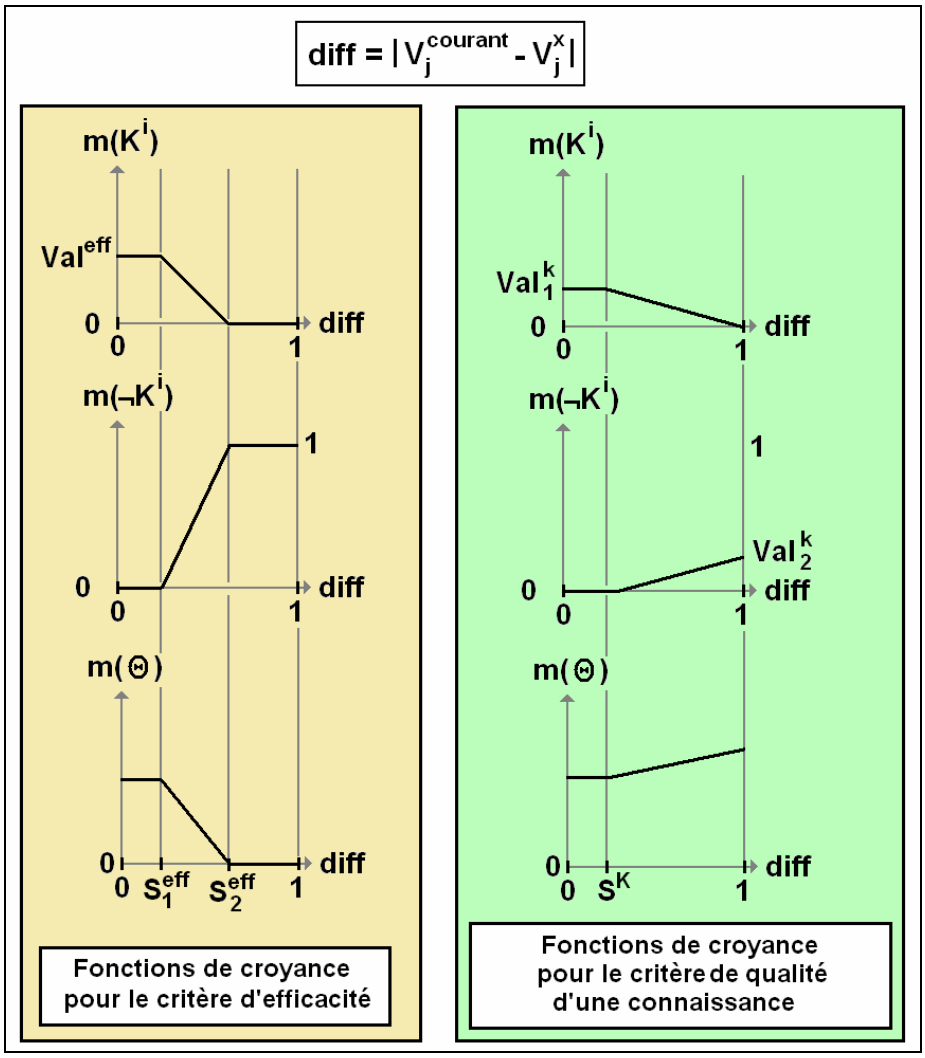


Figure E.5 Fonctions de croyance pour les critères d'efficacité et de qualité d'une connaissance

Soit $V^x = \{\text{Valeur}(V_j^{x'})\}_{j\in C}$, un vecteur de notes représentatif. Les masses de croyance pour les différents critères sont :

- Un *critère d'efficacité* :
  - Si l'écart entre la valeur du critère d'efficacité pour le vecteur de notes représentatif et sa valeur pour le vecteur de notes courant est inférieur au seuil S_1^{eff} , le critère indique :
    - que les deux vecteurs de notes doivent être appariés, avec une masse de croyance égale à Val^{eff},
    - que les deux vecteurs ne doivent pas être appariés, avec une masse de croyance égale à 0,
    - qu'il ne sait pas, avec une masse de croyance de : 1 - Val^{eff}.
  - Si l'écart entre les deux valeurs est compris entre S_1^{eff} et S_2^{eff} , alors :
    - la masse de croyance de la proposition « apparier les deux vecteurs de notes » diminue linéairement d'un facteur $(-Val^{eff})/(S_2^{eff} - S_1^{eff})$,
    - la masse de croyance de la proposition « ne pas apparier les deux vecteurs de notes » augmente linéairement d'un facteur $1/(S_2^{eff} - S_1^{eff})$.

- Si l'écart entre les deux valeurs est supérieur au seuil S_2^{eff} , alors :
  - la masse de croyance de la proposition « apparier les vecteur de notes » est égale à 0,
  - la masse de croyance de la proposition « ne pas apparier les deux vecteurs de notes » est égale à 1.

o Un *critère de qualité d'une connaissance* :

- Si l'écart entre la valeur du critère de qualité d'une connaissance k pour le vecteur de notes représentatif et sa valeur pour le vecteur de notes courant est inférieur au seuil S^k , le critère indique :
  - que les deux vecteurs de notes doivent être appariés, avec une masse de croyance égale à Val_1^k ,
  - que les vecteurs ne doivent pas l'être, avec une masse de croyance égale à 0,
  - qu'il ne sait pas, avec une masse de croyance de : 1 - Val_1^k .
- Si l'écart entre les deux valeurs est supérieur au seuil S^k , alors :
  - la masse de croyance de la proposition « apparier les deux vecteurs de notes » diminue linéairement d'un facteur (-Val_1^k)/(1 - S^k),
  - la masse de croyance de la proposition « ne pas apparier les deux vecteurs de notes » augmente linéairement d'un facteur Val_2^k /(1 - S^k).

Les fonctions de croyance que nous avons proposées nécessitent de définir différents paramètres : Val^{eff}, S_1^{eff} , S_2^{eff} et pour chaque connaissance k : Val_1^k , Val_2^k et S^k .

- Val^{eff} est défini tel que : $0 \leq Val^{eff} \leq 1$.
- S_1^{eff} et S_2^{eff} sont définis tels que : $0 \leq S_1^{eff} < S_2^{eff} \leq 1$.
- Val_1^k et Val_2^k sont définis tels que : $0 \leq Val_1^k < Val_2^k$ et tels que : $Val_1^k + Val_2^k \leq 1$.
- S^k est défini tel que : $0 \leq S^k \leq 1$.

La figure E.6 présente l'approche d'attribution de la catégorie de qualité pour le jeu de connaissances courant.

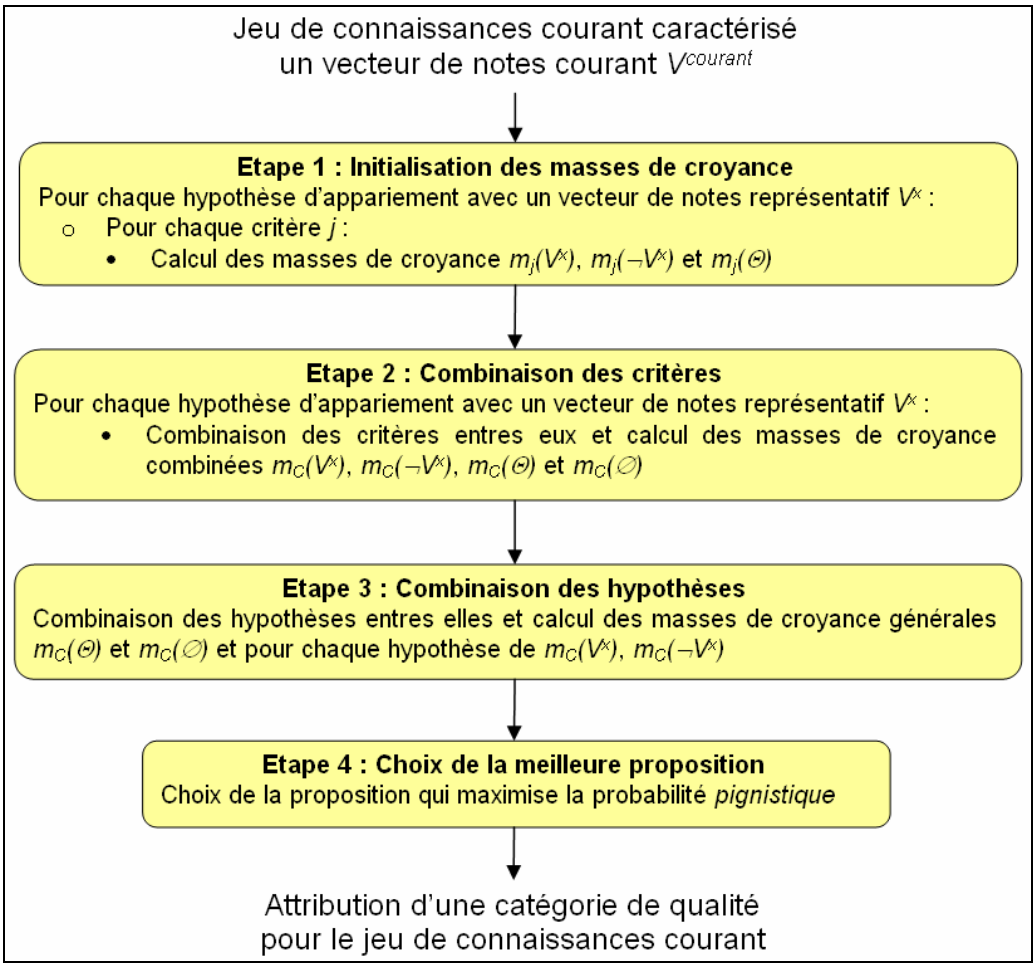


Figure E.6 Approche basée sur la théorie des fonctions de croyance pour l'attribution d'une catégorie de qualité pour un jeu de connaissances

Etape 1

La première étape de processus d'affectation d'une catégorie pour la théorie de l'évidence est la phase d'initialisation des masses de croyance. Cette phase consiste à calculer les masses de croyance pour le vecteur de notes courant par rapport à chaque vecteur de notes représentatif. Nous utilisons pour cela les fonctions présentées figure E.5. Pour un critère unique, la masse de croyance pour la proposition de conflit m(ø) est toujours égale à 0.

Le calcul des masses de croyance permet d'obtenir, pour chaque hypothèse d'appariement du jeu de connaissances courant avec un jeu de connaissances représentatif V^x, des résultats de la forme :

	$m(V^x)$	$m(\neg V^x)$	$m(\Theta)$	$m(\emptyset)$
Critère efficacité	0.5	0.3	0.2	0
Critère qualité connaissance k_1	0.2	0.05	0.75	0
Critère qualité connaissance k_2	0.1	0.1	0.8	0

Etape 2
Cette étape consiste à combiner les différents critères entre eux pour chaque hypothèse d'appariement du vecteur de notes courant avec un vecteur de notes représentatif V^x.

Nous nous intéressons pour cela aux intersections de propositions formulées par les différents critères.
Le tableau suivant donne le résultat des intersections entre deux propositions pour la fusion de deux critères :

$P = P' \cap P''$		**Critère C_2 P''**			
		$\mathbf{V^x}$	$\mathbf{\neg V^x}$	$\mathbf{\Theta}$	$\mathbf{\emptyset}$
Critère C_1 P'	$\mathbf{V^x}$	V^x	$\emptyset$	V^x	$\emptyset$
	$\mathbf{\neg V^x}$	$\emptyset$	$\neg V^x$	$\neg V^x$	$\emptyset$
	$\mathbf{\Theta}$	V^x	$\neg V^x$	Θ	$\emptyset$
	$\mathbf{\emptyset}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Pour donner un exemple de fonctionnement de ce tableau, nous nous intéresserons à la proposition V^x : celle-ci est obtenue après combinaison des critères C_1 et C_2 lorsque C_1 et C_2 donnent tous les deux V^x pour vraie et lorsque seul l'un des deux donne V^x pour vraie et que l'autre ne sait pas (ignorance, Θ).

Nous utilisons l'opérateur de fusion proposé par (Smets, 1990) pour calculer les masses de croyance résultant de la combinaison de deux critères :

$$\forall P \in 2^{\Theta}, m_{12}(P) = \sum_{P' \cap P'' = P} m_1(P') m_2(P'')$$

Cet opérateur permet d'obtenir les quatre masses de croyance résultant de la fusion des deux critères C_1 et C_2 : $(m_{12}(V^i), m_{12}(\neg V^i), m_{12}(\Theta), m_{12}(\emptyset))$.
Il est associatif et commutatif et permet donc de recombiner des masses de croyance déjà combinées à un troisième critère C_3 et d'obtenir ainsi quatre nouvelles masses de croyance : $(m_{123}(V^i), m_{123}(\neg V^i), m_{123}(\Theta), m_{123}(\emptyset))$.

Nous calculons ainsi les masses de croyance associées aux différentes propositions pour les cinq vecteurs de notes représentatifs. Nous obtenons ainsi, à la fin de cette étape les masses de croyance combinées $(m_C(V^x), m_C(\neg V^x), m_C(\Theta), m_C(\emptyset))$ pour chaque vecteur de notes représentatif V^x.

Etape 3
La troisième étape consiste à combiner les différentes hypothèses d'appariement du vecteur de notes courant avec un vecteur de notes représentatif V^x. L'intérêt de cette fusion est qu'elle permet d'intégrer dans la prise de décision finale le fait que les critères rejètent une hypothèse ($\neg V^x$).

De la même manière que pour la fusion des critères, nous allons utiliser le tableau des intersections suivant pour fusionner les hypothèses entre elles :

P = P'∩P''		**Vecteur de notes V^y P''**			
		$\mathbf{V^y}$	$\mathbf{\neg V^y}$	$\mathbf{\Theta}$	**ø**
Vecteur de notes V^x P'	$\mathbf{V^x}$	ø	V^x	V^x	ø
	$\mathbf{\neg V^x}$	V^y	$\{V^z\}_{V^z \in \{V^{rep}\} \wedge V^z \neq V^x \wedge V^z \neq V^y}$	$\neg V^x$	ø
	$\mathbf{\Theta}$	V^y	$\neg V^y$	Θ	ø
	ø	ø	ø	ø	ø

Nous utilisons pour cela l'opérateur de Dempster (Dempster, 1967) :

$$\forall P \in 2^{\Theta},\ m_{V^i,V^j}(P) = \frac{1}{1 - m_{V^i,V^j}(\phi)} \sum_{P' \cap P'' = P} m_{V^i}(P') m_{V^j}(P'')$$

Le coefficient $\frac{1}{1 - m_{V^i,V^j}(\phi)}$ sert à la normalisation de la masse de croyance obtenue. Dans le cas d'un conflit total ($m_{V^i,V^j}(\phi) = 1$), cette masse de croyance n'aura pas de sens et il ne sera pas possible de tirer une conclusion quant au meilleur appariement possible.

Cette formule combine chaque hypothèse d'appariement du vecteur de notes courant avec un vecteur de notes représentatif V^i et obtenons au final une masse de croyance pour chaque proposition ($m(V^{Mv})$, $m(\neg V^{Mv})$, $m(V^{My})$, $m(\neg V^{My})$, $m(\Theta)$, etc.).

Etape 4

Cette dernière étape consiste à déterminer parmi les propositions $\{V^{TMv}\}$, $\{V^{Mv}\}$, $\{V^{M}\}$, $\{V^{Bn}\}$ et $\{V^{TBn}\}$ celle qui a le plus de chances d'être vraie.

Pour déterminer quelle est la meilleure proposition, nous proposons d'utiliser la probabilité pignistique (Smets & Kennes, 1994). Cette probabilité est très largement utilisée pour les problèmes tels que le nôtre où l'on souhaite sélectionner une hypothèse simple et non un ensemble d'hypothèses. En effet, nous souhaitons dans notre cas apparier le vecteur de notes courant à un seul vecteur de notes représentatif.

La probabilité pignistique d'une proposition A est donnée par la formule suivante :

$$P(A) = \sum_{A \subseteq B} m(B) \frac{|A|}{|B|}$$

A travers cette application, nous proposons de ne sélectionner qu'un seul jeu de connaissances. Nous calculons donc cette probabilité uniquement pour les propositions $\{V^{TMv}\}$, $\{V^{Mv}\}$, $\{V^{M}\}$, $\{V^{Bn}\}$ et $\{V^{TBn}\}$. Nous rappelons que la proposition Θ est égale à $\{V^{TMv}, V^{Mv}, V^{My}, V^{Bn}, V^{TBn}\}$ et que la proposition $\{\neg V^{TBn}\}$ est égale à $\{V^{TMv}, V^{Mv}, V^{My}, V^{Bn}\}$.

Si les masses de croyance obtenues après combinaison des critères sont de la forme :

	$\mathbf{m(V^x)}$	$\mathbf{m(\neg V^x)}$	$\mathbf{m(\Theta)}$
$\{V^{TMv}\}$	0	0.2	0.1
$\{V^{Mv}\}$	0	0.1	
$\{V^{My}\}$	0.05	0.15	
$\{V^{Bn}\}$	0.3	0	
$\{V^{TBn}\}$	0.1	0	

Les probabilités pignistique obtenues sont :

- $P(\{V^{TMv}\}) = \frac{1}{1}\times 0 + \frac{1}{5}\times(0.1) + \frac{1}{4}\times(0.1+0.15+0+0) = 0.0825$
- $P(\{V^{Mv}\}) = 0.1075$
- $P(\{V^{My}\}) = 0.145$
- $P(\{V^{Bn}\}) = 0.4325$
- $P(\{V^{TBn}\}) = 0.2325$

Le vecteur de notes courant est donc apparié au vecteur de notes représentatif V^{Bn} et la qualité du jeu de connaissances est donc affectée à la catégorie « Bonne ».

Nous obtenons donc à la fin de cette étape une affectation de catégorie pour le jeu de connaissances courant.

L'utilisation de cette méthode implique de définir de nombreux paramètres ainsi que de choisir entre deux procédures. Il est en effet nécessaire de définir les paramètres Val^{eff}, S_1^{eff} et S_2^{eff} pour le critère de validité et Val_1^k, Val_2^k et S^k pour chaque connaissance *k*.
Un point important d'une prise de décision concerne la robustesse de la décision prise (Roy, 1997), c'est-à-dire sa sensibilité aux variations de valeur des paramètres. Une approche pour tester la robustesse d'une décision consiste à analyser ses variations quand les différents paramètres varient.
Nous proposons donc, pour notre application, de tester notre méthode basée sur la théorie des fonctions de croyance avec différents jeux de paramètres et de décider de l'affectation finale par vote majoritaire (figure E.7). Ainsi, la catégorie obtenant le plus de votes sera toujours retenue. Nous disposerons en plus d'un indicateur de fiabilité de la décision (le taux de confiance) qui dépendra de l'écart par rapport aux nombres de votes recueillis par les autres catégories.

La figure E.7 illustre notre approche : à chaque fois que notre méthode basée sur la théorie des fonctions de croyance est utilisée pour un jeu de paramètres, elle propose en sortie une catégorie, parmi l'ensemble *Cat* des catégories possibles, pour la procédure pessimiste et une

pour la procédure optimiste (les deux catégories peuvent être les mêmes). Nous notons *Aff(A)* l'ensemble des sorties obtenues pour un vecteur de notes courant *A* et $nombre_x(Aff(A))$ le nombre d'affectations de la catégorie *x* au vecteur de notes *A*. La catégorie affectée en fin de processus d'évaluation est celle qui est la plus représentée dans *Aff(A)*. L'approche fournit un taux de confiance compris entre 0 et 1. Ce taux exprime la robustesse de l'affectation choisie. Cette approche nous permet d'obtenir non seulement une affectation finale plus robuste mais également un taux de confiance pour cette affectation.

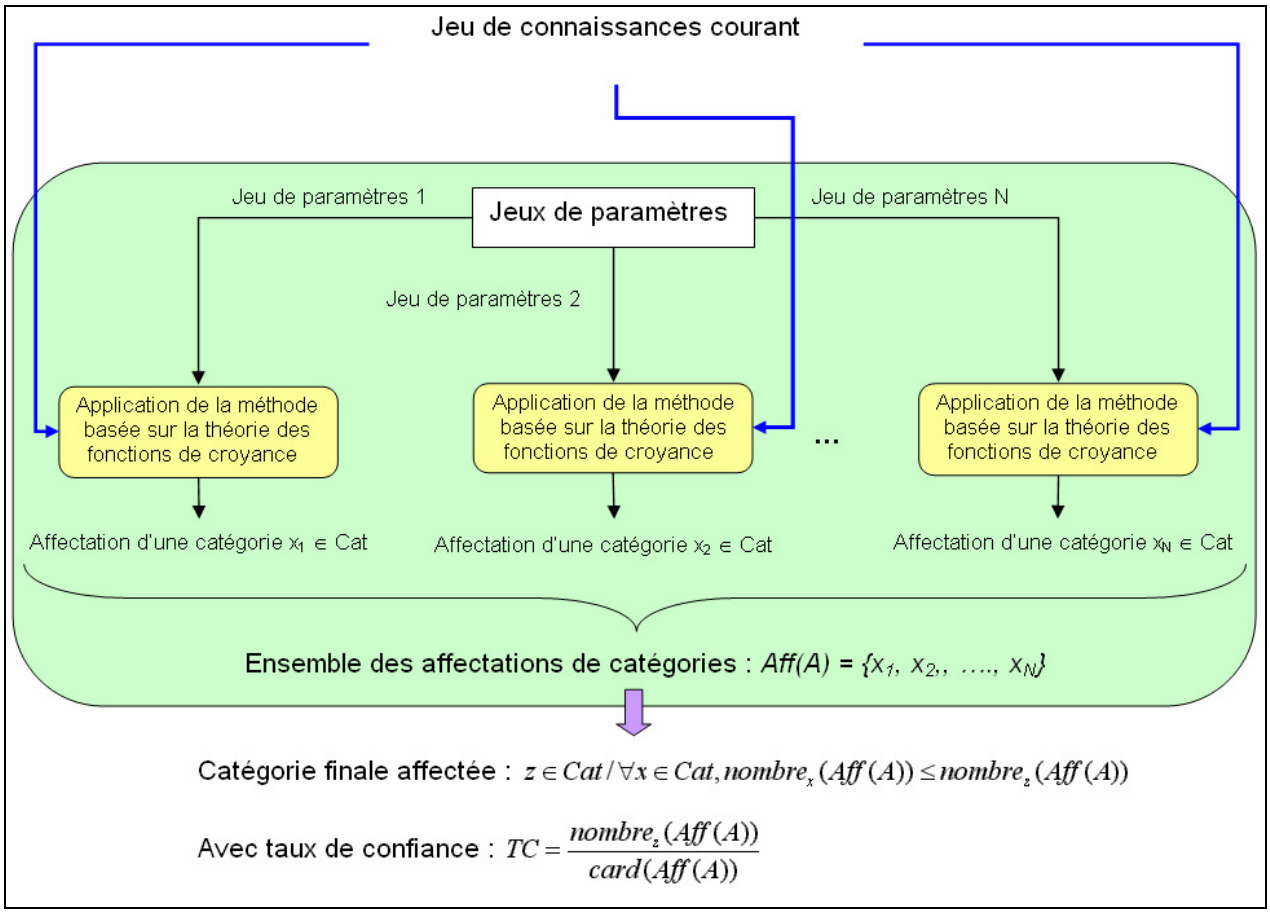


Figure E.7 Approche générale d'affectation d'une catégorie avec système de votes par la méthode basée sur la théorie des fonctions de croyance

Un point important de cette approche réside dans le choix des différentes valeurs de paramètres à tester. Ces valeurs doivent toujours rester pertinentes. L'objectif n'est pas de tester les variations de résultats avec des valeurs extrêmement différentes mais juste de tester si de légères variations des valeurs des paramètres entraînent ou non des changements sur la catégorie affectée.

E.III.4.2.3 Evaluation du jeu de connaissances par la méthode ELECTRE TRI

Dans la partie précédente (E.III.4.2.2), nous avons proposé d'utiliser une approche basée sur la théorie des fonctions de croyance pour notre problème d'affectation d'une catégorie au jeu de connaissances courant. Dans cette partie E.III.4.2.3, nous proposons de recourir à une autre méthode basée cette fois sur la méthode ELECTRE TRI (Yu, 1992).

Cette méthode fait partie de la famille des méthodes ELECTRE. Elle a été utilisée avec succès pour de nombreuses applications (Raju et al., 2000 ; Georgopoulou et al., 2003 ; Lourenço & Costa, 2004).
La méthode ELECTRE TRI est particulièrement bien adaptée à notre problématique d'évaluation des connaissances. En effet, dans leur présentation des problèmes pour lesquels les méthodes ELECTRE sont appropriées, (Figueira et al., 2005) mettent en avant les points suivants :

- *Le nombre de critères est supérieur à trois* : nous avons dans notre problème un critère par connaissance, plus un pour l'efficacité du système de résolution de problèmes. Le nombre de critères sera donc très généralement supérieur à trois.
- *Les critères sont de nature hétérogène* : nous avons dans notre problème des critères de qualité des connaissances et un critère d'efficacité. Or, il peut s'avérer délicat de comparer directement le critère d'efficacité à un critère de qualité d'une connaissance.
- Nous souhaitons intégrer dans la prise de décision le fait que pour certains critères, une petite différence de valeur n'est pas significative alors que l'accumulation de petites différences peut devenir significative : nos critères de qualité des connaissances étant peu fiables, il peut être intéressant de considérer que seuls des écarts suffisamment importants entre valeurs pour ces critères sont significatifs.

Le principe de cette seconde méthode d'affectation d'une catégorie au jeu de connaissances courant est de comparer le vecteur de notes courant aux quatre vecteurs de notes *frontière* (cf. E.III.4.1). Nous cherchons, en effet, à déterminer les relations existant entre le vecteur de notes courant et les vecteurs de notes *frontière*. Nous nous intéressons particulièrement à la relation de surclassement (***S***). La notation a***S***b indique que le vecteur de notes *a* surclasse le vecteur de notes *b*, c'est-à-dire qu'il est au moins aussi bon que lui.

L'application de la méthode ELECTRE TRI nécessite de choisir différents paramètres pour chaque critère :

- son poids (w) : il traduit l'importance du critère dans l'affectation d'une catégorie,
- le seuil de préférence (p) : il traduit le seuil à partir duquel la différence entre deux valeurs pour ce critère permet de préférer un vecteur de notes à un autre. Lorsque la différence entre deux valeurs pour ce critère est supérieure à ce seuil, le vecteur de notes qui a la note la plus élevée est considéré comme de meilleure qualité pour ce critère,
- le seuil d'indifférence (q) : il traduit le seuil à partir duquel la différence entre deux valeurs pour ce critère est considérée comme significative. Lorsque la différence entre deux valeurs pour ce critère est inférieure à ce seuil, les deux vecteurs de notes sont supposés de même qualité pour ce critère,
- le seuil de veto (v) : il traduit le seuil à partir duquel la différence entre deux valeurs pour ce critère disqualifie le vecteur de notes avec la plus petite valeur. Lorsque la différence entre deux valeurs pour ce critère est supérieure à ce seuil, le vecteur de notes de valeur la plus basse ne pourra jamais être considéré comme de meilleure qualité.

Un dernier paramètre est le seuil de coupe (λ). Ce paramètre définit le seuil de référence pour la comparaison entre deux vecteurs de notes. Plus ce seuil est élevé, plus, lors de la comparaison d'un vecteur de notes V_1 avec un vecteur de notes V_2, les critères devront être unanimes quant à la supériorité de V_1 sur V_2 pour que le vecteur de notes V_1 soit supposé surclasser V_2.

La figure E.8 présente les différentes étapes de la méthode ELECTRE TRI. $V^{courant}$ est un vecteur de notes représentant le jeu de connaissances courant.

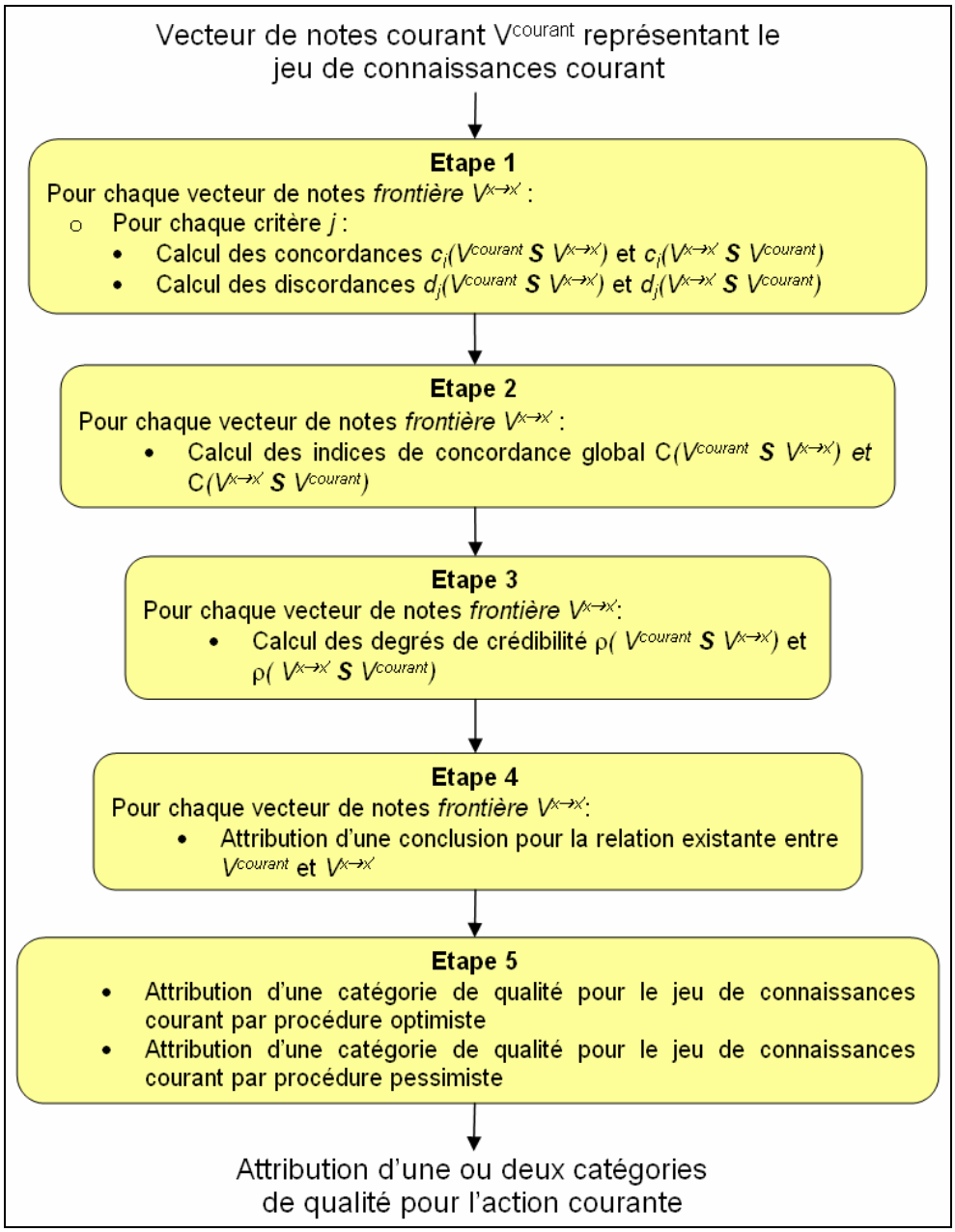


Figure E.8 Méthode ELECTRE TRI appliquée à l'évaluation de la qualité d'un jeu de connaissances

Etape 1

La première étape de la méthode ELECTRE TRI consiste à calculer, pour chaque vecteur de notes *frontière* $V^{x\to x'}$ et pour chaque critère j, les concordances $c_j(V^{courant}\ \boldsymbol{S}\ V^{x\to x'})$ et $c_j(V^{x\to x'}\ \boldsymbol{S}\ V^{courant})$ entre le vecteur de notes courant et ce vecteur de notes *frontière* ainsi que les discordances $d_j(V^{courant}\ \boldsymbol{S}\ V^{x\to x'})$ et $d_j(V^{x\to x'}\ \boldsymbol{S}\ V^{courant})$.

La concordance $c_j(a\ \boldsymbol{S}\ b)$ traduit le niveau de certitude, du critère j envers l'assertion que le vecteur de notes a est au moins aussi bon que le vecteur de notes b et la discordance $d_j(a\ \boldsymbol{S}\ b)$ traduit le niveau de certitude pour j, que ce n'est pas le cas.

La figure E.9 illustre la façon de calculer ces valeurs, pour deux vecteurs de notes *a* et *b*, et un critère *j*. Nous notons *valeur(a_j)* la valeur du critère *j* pour le vecteur de notes *a*. Les valeurs de concordance et de discordance dépendent des paramètres définis pour chaque critère (des seuils de préférence (*p*), d'indifférence (*q*) et de veto (*v*)).

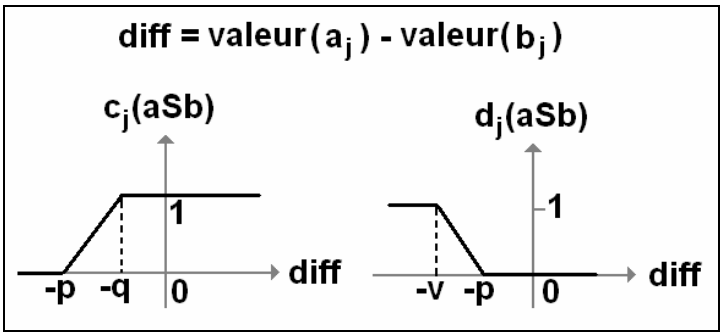


Figure E.9 Concordance et discordance

Etape 2

La seconde étape consiste à calculer, pour chaque vecteur de notes *frontière* $V^{x \to x'}$, *les indices* de *concordance globaux* $C(V^{courant}\ \boldsymbol{S}\ V^{x \to x'})$ et $C(V^{x \to x'}\ \boldsymbol{S}\ V^{courant})$ entre le vecteur de notes courant $V^{courant}$ et ce vecteur de notes *frontière*. Cet indice représente la concordance moyenne obtenue pour l'ensemble des critères pondérée par leurs poids. Il permet d'estimer la part de critères indiquant qu'un vecteur de notes est au moins aussi bon (surclasse) qu'un autre. Nous notons w_j le poids d'un critère *j*. Pour deux vecteurs de notes *a* et *b*, l'indice de concordance global *C(a**S**b)* est donné par la formule suivante :

$$C(aSb) = \frac{\sum_{j \in C} w_j \times c_j(aSb)}{\sum_{j \in C} w_j}$$

Etape 3

La troisième étape consiste à calculer, pour chaque vecteur de notes *frontière* $V^{x \to x'}$, les *degrés de crédibilité* du surclassement global $\rho\,(V^{courant}\ \boldsymbol{S}\ V^{x \to x'})$ et $\rho\,(V^{x \to x'}\ \boldsymbol{S}\ V^{courant})$ entre le vecteur de notes courant $V^{courant}$ et ce vecteur de notes *frontière*. La fonction $\rho(a\boldsymbol{S}b)$ exprime le degré avec lequel le vecteur de notes *a* est a priori au moins aussi bon que le vecteur *b*. Elle correspond à *l'indice de concordance global* affaibli par les possibles effets de veto. En effet, soient *a* et *b* deux vecteurs de notes. Si la différence de valeur entre *a* et *b* pour un critère dépasse la valeur de veto de ce critère, la discordance pour ce critère vaut automatiquement 1 ($d_j(a\boldsymbol{S}b) = 1$), et le *degré de crédibilité* $\rho(a\boldsymbol{S}b)$ vaut 0 : le vecteur de notes *a* n'est jamais considéré comme aussi bon que le vecteur *b*.

$$\rho(aSb) = C(aSb) \times \prod_{j \in \{i \in C / d_i(aSb) > C(aSb)\}} \frac{1 - d_j(aSb)}{1 - C(aSb)}$$

Etape 4

La quatrième étape consiste, à partir des *degrés de crédibilité*, à établir la relation existant entre le vecteur de notes courant $V^{courant}$ et chaque action de référence $V^{x \to x'}$ (figure E.10). Quatre types de relations sont possibles :

- $V^{courant}$ ***R*** $V^{x \to x'}$: signifie que le vecteur de notes $V^{courant}$ et le vecteur de notes $V^{x \to x'}$ sont incomparables. Cette relation unit deux vecteurs de notes où aucun des deux ne surclasse l'autre : il n'est donc pas possible de les comparer.
- $V^{x \to x'}$ ***S*** $V^{courant}$: signifie que le vecteur de notes $V^{courant}$ surclasse le vecteur de notes $V^{x \to x'}$, c'est-à-dire qu'il est au moins aussi bon que lui.
- $V^{courant}$ ***S*** $V^{x \to x'}$: signifie que le vecteur de notes $V^{x \to x'}$ surclasse le vecteur de notes $V^{courant}$.
- $V^{courant}$ ***I*** $V^{x \to x'}$: signifie que les vecteurs de notes $V^{courant}$ et $V^{x \to x'}$ sont de même qualité. Cette relation unit deux vecteurs de notes qui se surclassent mutuellement : ils sont donc tous les deux aussi bons.

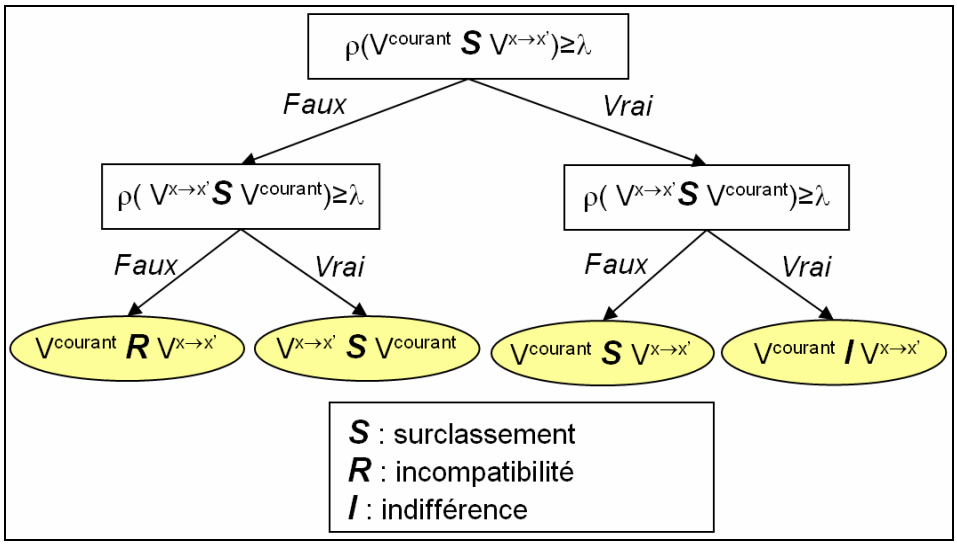


Figure E.10 Relation entre un vecteur de notes $V^{courant}$ *A* et un vecteur de notes frontière $V^{x \to x'}$

Etape 5

La dernière étape consiste à attribuer une catégorie au jeu de connaissances courant représenté par le vecteur de notes courant $V^{courant}$. Il existe deux procédures possibles pour affecter une catégorie à un jeu de connaissances :

- La procédure *pessimiste* :
  - si $V^{courant}$ ***S*** $V^{Bn \to TBn}$, alors le jeu de connaissances courant appartient à la catégorie de qualité « très bonne »
  - sinon si $V^{courant}$ ***S*** $V^{My \to Bn}$, alors le jeu de connaissances courant appartient à la catégorie de qualité « bonne »
  - sinon si $V^{courant}$ ***S*** $V^{Mv \to My}$, alors le jeu de connaissances courant appartient à la catégorie de qualité « moyenne »
  - sinon si $V^{courant}$ ***S*** $V^{TMv \to Mv}$, alors le jeu de connaissances courant appartient à la catégorie de qualité « mauvaise »
  - sinon le jeu de connaissances courant appartient à la catégorie de qualité « très mauvaise »

- La procédure *optimiste* :
  - si $\neg(V^{Bn \to TBn}\ \boldsymbol{S}\ V^{courant})$, alors le jeu de connaissances courant appartient à la catégorie de qualité « très bonne »
  - sinon si $\neg(V^{My \to Bn}\ \boldsymbol{S}\ V^{courant})$, alors le jeu de connaissances courant appartient à la catégorie de qualité « bonne »
  - sinon si $\neg(V^{My \to My}\ \boldsymbol{S}\ V^{courant})$, alors le jeu de connaissances courant appartient à la catégorie de qualité « moyenne »
  - sinon si $\neg(V^{My \to TMy}\ \boldsymbol{S}\ V^{courant})$ alors le jeu de connaissances courant appartient à la catégorie de qualité « mauvaise »
  - sinon le jeu de connaissances courant appartient à la catégorie de qualité « très mauvaise »

La différence entre les deux procédures provient de la gestion des relations *indifférence* (***I***) et *incompatibilité* (***R***). Dans le cas de la procédure *pessimiste*, lorsque le vecteur de notes courant et l'un des vecteurs de notes *frontière* sont *incompatibles* ou *indifférents*, le vecteur de notes courant est considéré comme inférieur, alors qu'il est considéré comme supérieur pour la procédure *optimiste*.

L'utilisation de la méthode ELECTRE TRI nécessite, comme la méthode basée sur les fonctions de croyance, de définir de nombreux paramètres. Elle nécessite également de choisir entre deux procédures. Nous proposons, comme pour notre première approche d'affectation de catégorie, d'améliorer la robustesse de la décision en testant différents jeux de paramètres (cf. E.III.4.2.2). La catégorie finale affectée sera celle obtenant le plus de votes (figure E.11). L'approche fournit un taux de confiance compris entre 0 et 1, qui exprime la robustesse de l'affectation choisie. Comme pour la méthode basée sur les fonctions de croyance, les jeux de paramètres testés doivent être pertinents. L'objectif n'est pas de tester des valeurs de paramètres très différentes mais de vérifier si les légères variations de valeur des paramètres entraînent des modifications dans la catégorie affectée.

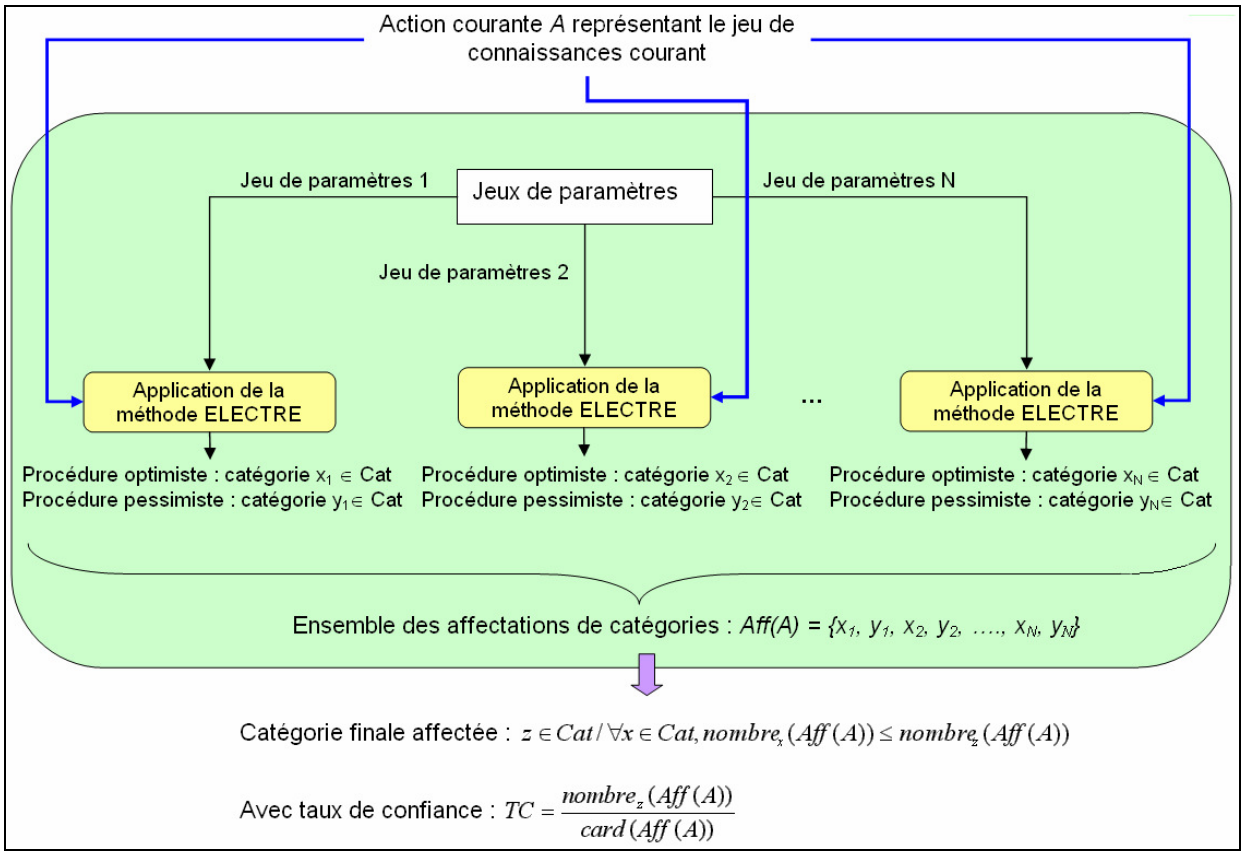


Figure E.11 Approche générale d'affectation d'une catégorie avec système de votes par la méthode ELECTRE TRI

E.III.4.2.4 Conclusion sur l'évaluation qualitative du jeu de connaissances

Nous avons proposé deux méthodes pour résoudre notre problème d'évaluation qualitative du jeu de connaissances courant : une méthode basée sur la méthode ELECTRE TRI et une méthode basée sur la théorie des fonctions de croyance. Ces deux méthodes ont été utilisées avec succès dans de nombreux domaines. Elles paraissent particulièrement adaptées à notre problème dans lequel les critères de décision sont peu fiables.
Nous proposons, au chapitre F, une expérimentation menée dans le cadre du diagnostic de la qualité des connaissances du modèle AGENT comparant les résultats obtenus avec ces deux approches.

Nous obtenons, à la fin de cette phase d'évaluation qualitative du jeu de connaissances, une catégorie de qualité pour caractériser la qualité du jeu de connaissances courant. Si cette catégorie est « moyenne », « mauvaise » ou « très mauvaise » *l'agent diagnostic* proposera à l'utilisateur de déclencher un processus de révision. Il présentera également à l'utilisateur une évaluation de la qualité de chaque connaissance. Nous détaillons le principe de cette évaluation dans la partie suivante.

E.III.4.3 Evaluation indépendante de la qualité des connaissances

Nous avons introduit des méthodes pour évaluer qualitativement la qualité d'un jeu de connaissances. Lorsque celui-ci est de qualité moyenne, mauvaise ou très mauvaise, l'agent *diagnostic* peut proposer à l'utilisateur de déclencher un processus de révision.

De plus, il peut fournir un bilan sur la qualité de chacune des connaissances. L'enjeu est de permettre à l'utilisateur de savoir exactement quelles connaissances posent a priori des problèmes et quelles connaissances fonctionnent bien au vu de l'expérience.

Nous avons introduit, en partie E.III.4.2, des catégories pour évaluer qualitativement la qualité d'un jeu de connaissances. Nous proposons de reprendre exactement les mêmes catégories pour évaluer chacune des connaissances et d'utiliser la note (le critère) donnée à chaque connaissance pour lui attribuer une catégorie. Ainsi, un agent *connaissance* donnant une note comprise entre $Valeur(K^{Mv \to My})$ et $Valeur(K^{My \to Bn})$ pour la connaissance qu'il représente pourra affecter la catégorie de qualité « moyenne » à sa connaissance.

Un problème existant, déjà évoqué en partie E.II.1, concerne l'absence de contrôle de l'efficacité du système de résolution de problèmes dans le cadre de l'évaluation de chaque connaissance. Ainsi, un jeu de connaissances peut être composé de connaissances qui semblent chacune parfaite, mais très mauvaises en réalité car ne permettant pas au système d'obtenir des résultats satisfaisants en termes de qualité.

Un premier indicateur pour évaluer la fiabilité de l'évaluation faite de chaque connaissance est donc la note d'efficacité. Un utilisateur obtenant, pour son jeu de connaissances, une note d'efficacité très mauvaise sait que, quelles que soient les évaluations obtenues par les connaissances, celles-ci nécessitent d'être révisées.

Un second indicateur permettant à l'utilisateur de pouvoir quantifier la fiabilité des évaluations obtenues est le *taux moyen d'états utiles visités*. En effet, un taux très élevé indique que le système n'a pas beaucoup développé les arbres d'états et donc n'a eu que peu d'opportunités de trouver de meilleurs états.

E.III.5 Bilan sur l'approche de diagnostic proposée

Nous avons détaillé, dans cette partie E.III, l'approche de diagnostic en ligne de la qualité des connaissances que nous proposons.

Notre approche de diagnostic est basée sur l'analyse des instances résolues du problème considéré. A chaque résolution d'une instance par le système de résolution de problèmes, les *agents connaissance* analysent les succès et les échecs rencontrés par leurs connaissances respectives et *l'agent diagnostic* mémorise la qualité du meilleur état obtenu. Une fois suffisamment d'instances résolues, les *agents connaissance* et *l'agent diagnostic* donnent une note au critère qu'ils représentent. *L'agent diagnostic* établit ensuite une évaluation qualitative de la qualité globale du jeu de connaissances courant à partir de ces notes. Nous avons présenté, dans ce cadre, deux méthodes de décision multicritère que nous avons proposé d'utiliser pour notre problème d'évaluation globale du jeu de connaissances courant. La première approche est basée sur la théorie des croyances et la seconde approche est basée sur la méthode ELECTRE TRI.

L'évaluation globale obtenue permet à *l'agent diagnostic* de décider s'il faut ou non proposer à l'utilisateur de réviser les connaissances. Il peut également fournir une évaluation qualitative de la qualité de chacune des connaissances.

Nous détaillons dans la partie suivante (E.IV) la mise en œuvre notre approche de diagnostic pour le cas du modèle AGENT.

E.IV Application pour le modèle AGENT

E.IV.1 Les agents connaissance et l'agent diagnostic

Nous rappelons que le modèle AGENT, tel que nous l'avons enrichi, comprend six types de connaissances différents : les connaissances de priorité des contraintes, les connaissances d'application des actions, les connaissances de restriction des actions, le critère de fin de cycle, le critère de validité des états et le critère d'optimalité des états.

Nous rappelons également que chaque connaissance est représentée par un *agent connaissance* et qu'un *agent diagnostic* est défini par classe *d'agents géographiques*.

Durant le processus de diagnostic, chaque *agent connaissance* analyse la qualité de la connaissance qu'il représente et fournit ainsi une note traduisant la qualité du critère qu'il représente.

Si l'on considère un *agent géographique* ayant M contraintes et pouvant s'appliquer N actions, le nombre *d'agents connaissance* est de : $M + 2\times N + 3$. Pour un exemple classique *d'agent géographique* disposant de 4 contraintes et pouvant s'appliquer 5 actions, ce nombre s'élève à 17. Ce qui fait, si l'on ajoute le critère d'efficacité (représenté par *l'agent diagnostic*), 18 critères.

E.IV.2 Analyse du résultat d'une généralisation d'un agent géographique

A chaque généralisation, les agents *connaissance* et l'agent *diagnostic* analysent l'arbre d'états obtenu. L'agent *diagnostic* mémorise la satisfaction du meilleur état trouvé ainsi que le taux d'états utiles. Les agents *connaissance*, pour leur part, analysent les arbres d'états afin de calculer leurs nombres de succès et d'échecs. L'enjeu est de comparer le contenu du jeu de connaissances courant avec des faits tirés de l'expérience (les arbres d'états). Nous rappelons que les arbres d'états ont été construits ici avec le jeu de connaissances courant et non avec un jeu de connaissances *minimal*. Ils seront donc en général beaucoup plus élagués que ceux utilisés dans le cadre du processus de révision des connaissances.

Comme nous l'avons évoqué en partie E.III.3, une première étape de l'analyse d'un arbre d'états consiste à construire la liste des meilleurs chemins. Une fois cette liste construite, les différents agents *connaissance* calculent les statistiques pour leurs connaissances.

Nous rappelons que les succès et les échecs peuvent être de deux types.
Pour les succès :

- *vrais positifs* : cas où *l'agent connaissance* a pensé à juste titre que le concept qu'il représente était vrai,
- *vrais négatifs* : cas où *l'agent connaissance* a pensé que le concept qu'il représente était faux.

Pour les échecs :

- *faux positifs* : cas où *l'agent connaissance* a pensé que le concept qu'il représente était vrai alors qu'il ne l'était pas,
- *faux négatifs* : cas où *l'agent connaissance* a pensé que le concept qu'il représente était faux alors qu'il ne l'était pas.

Le calcul de ces statistiques ainsi que leur signification dépend du type de connaissance :

- ***Connaissance d'application d'une action*** : concept représenté : « *il faut appliquer l'action* »
  - *Faux positif* : cas où l'action est appliquée mais mène à un état inutile (n'appartenant pas à un meilleur chemin).
  - *Vrai positif* : cas où l'action est appliquée et mène à un état utile.
  - *Vrai négatif* : cas où l'action n'a pas été appliquée.
  - *Faux négatif* : cas où l'action est appliquée et mène à un état utile mais où l'action n'a pas été appliquée en priorité par rapport aux autres actions proposées par la même contrainte.

- ***Connaissance de restriction d'une action*** : concept représenté : « *il ne faut pas appliquer l'action* (*le nombre limite d'application est atteint)* »
  - *Faux positif* : pas de faux positif. En effet, contrairement au cadre de la révision des connaissances où l'utilisation d'un jeu de connaissances *minimal* lors de la construction des arbres nous permettait d'analyser si le fait de ne pas appliquer une action était une bonne ou une mauvaise chose, on ne peut savoir ce qui se serait passé si la limite d'application de l'action n'avait pas été atteinte et si l'action avait pu être appliquée. A chaque fois que la restriction sera atteinte nous serons donc dans le cas d'un vrai positif.
  - *Vrai positif* : cas où le nombre limite d'application est bien choisi (il est égal au nombre de fois où il fut nécessaire d'appliquer l'action pour arriver au meilleur état).
  - *Vrai négatif* : cas où l'action est appliquée et mène à un état utile (il fut judicieux de permettre à l'action de pouvoir s'appliquer une fois de plus)
  - *Faux négatif* : cas où la restriction est supérieure au nombre de fois où l'action a été proposée pour arriver au meilleur état.

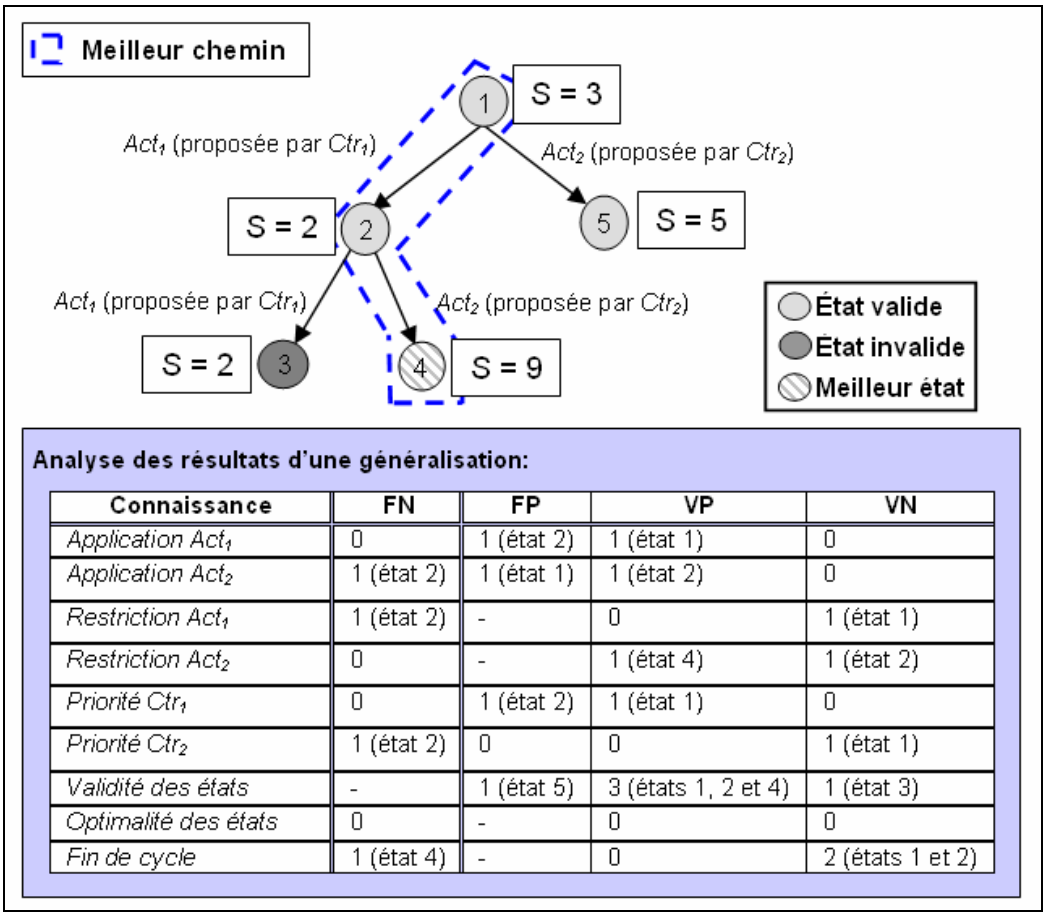
- ***Connaissance de priorité d'une contrainte*** : concept représenté : « *il faut que la contrainte soit prioritaire »*
  - *Faux positif* : cas où la contrainte est désignée comme prioritaire mais où toutes les actions qu'elle a proposées mènent à des états inutiles.
  - *Vrai positif* : cas où la contrainte est désignée comme prioritaire et où l'une des actions qu'elle a proposées mène à un état utile.
  - *Vrai négatif* : cas où la contrainte n'est pas désignée comme prioritaire et où toutes les actions qu'elle a proposées mènent à des états inutiles.
  - *Faux négatif* : cas où la contrainte n'est pas désignée comme prioritaire mais où l'une des actions qu'elle a proposées mène à un état utile.

- ***Critère de validité des états*** : concept représenté : « *l'état doit être valide* »
  - *Faux positif* : cas où un état inutile est valide.
  - *Vrai positif* : cas où un état inutile est invalide.
  - *Vrai négatif* : cas où un état inutile est valide.
  - *Faux négatif* : pas de faux négatif : on ne peut pas savoir si un état invalide aurait pu être utile s'il avait été valide. Le cas d'un état invalide est donc toujours considéré comme un vrai négatif.

- ***Critère d'optimalité des états*** : concept représenté : « *l'état doit être optimal* »
  - *Faux positif* : pas de faux positif : on ne peut pas savoir si un état optimal aurait toujours été optimal si on avait pu continuer l'exploration à partir de celui-ci. Le cas d'un état optimal est donc toujours considéré comme un vrai positif.
  - *Vrai positif* : cas d'un état optimal.
  - *Vrai négatif* : cas d'un état valide non optimal dont l'un des descendants a une satisfaction supérieure.
  - *Faux négatif* : cas d'un état valide non optimal dont aucun des descendants n'a de satisfaction supérieure.

- ***Critère de fin de cycle*** : concept représenté : « *l'exploration doit s'arrêter* »
  - *Faux positif* : pas de faux positif : on ne peut pas savoir s'il aurait été possible de trouver un meilleur état s'il avait été possible de poursuivre l'exploration. Un cas où la connaissance relative à la fin de cycle entraîne l'arrêt de l'exploration de l'arbre d'états est toujours considéré comme un vrai positif.
  - *Vrai positif* : cas où la connaissance relative à la fin de cycle entraîne l'arrêt de l'exploration de l'arbre d'états.
  - *Vrai négatif* : cas d'un état utile qui n'est pas le meilleur état.
  - *Faux négatif* : cas où la connaissance n'a pas entraîné l'arrêt de l'exploration de l'arbre d'états et où le meilleur état n'est pas un état parfait.

La figure E.12 présente un exemple de statistiques obtenues après analyse d'un arbre d'états.



Analyse des résultats d'une généralisation:

Connaissance	FN	FP	VP	VN
Application Act_1	0	1 (état 2)	1 (état 1)	0
Application Act_2	1 (état 2)	1 (état 1)	1 (état 2)	0
Restriction Act_1	1 (état 2)	-	0	1 (état 1)
Restriction Act_2	0	-	1 (état 4)	1 (état 2)
Priorité Ctr_1	0	1 (état 2)	1 (état 1)	0
Priorité Ctr_2	1 (état 2)	0	0	1 (état 1)
Validité des états	-	1 (état 5)	3 (états 1, 2 et 4)	1 (état 3)
Optimalité des états	0	-	0	0
Fin de cycle	1 (état 4)	-	0	2 (états 1 et 2)

Figure E.12 Exemple de statistiques obtenues pour un agent géographique

- ***Connaissance d'application d'une action*** :
  - *Etat 1* : l'application de Act_1 a permis de rester sur le meilleur chemin contrairement à l'application de Act_2. La connaissance d'application de Act_1 admet donc un vrai positif et celle de Act_2 un faux positif.
  - *Etat 2* : l'application de Act_2 a permis de rester sur le meilleur chemin contrairement à l'application de Act_1. La connaissance d'application de Act_2 admet donc un vrai positif et celle de Act_1 un faux positif.

- ***Connaissances de restriction d'une action*** :
  - Pour arriver au meilleur état, l'action Act_1 et l'action Act_2 n'auraient dû être proposées qu'une fois chacune. Nous posons que dans le jeu de connaissances la restriction d'application pour Act_1 est égale à 3 et celle pour Act_2 à 1. Dans ce cas, Act_1 admet un faux négatif (la connaissance aurait dû limiter le nombre d'applications à 1) et un vrai négatif (la connaissance a bien permis que l'action soit appliquée), et Act_2 un vrai positif (la connaissance a bien limité le nombre d'applications à 1) et un vrai négatif (la connaissance a bien permis que l'action soit appliquée).

- ***Connaissance de priorité des contraintes*** :
  - *Etat 1* : l'application de Act_1 proposée par Ctr_1 a permis de rester sur le meilleur chemin contrairement à l'application de Act_2 proposée par Ctr_2. La contrainte Ctr_1 a la priorité pour cet état sur la contrainte Ctr_2. La connaissance de priorité de Ctr_1 admet donc un vrai positif, et celle de Ctr_2, un vrai négatif.
  - *Etat 2* : l'application de Act_2 proposée par Ctr_2 a permis de rester sur le meilleur chemin contrairement à l'application de Act_1 proposée par Ctr_1. La contrainte Ctr_1 a la priorité pour cet état sur la contrainte Ctr_2. La connaissance de priorité de Ctr_1 admet donc un faux positif, et celle de Ctr_2, un faux négatif.

- ***Connaissance de validité des états*** :
  - *Etats 1, 2 et 4* : Ces états appartiennent au meilleur chemin (et sont donc valides). La connaissance de validité admet donc un vrai positif pour chacun.
  - *Etat 3* : Cet état est invalide alors que son prédécesseur appartient à un meilleur chemin. La connaissance de validité admet donc un vrai négatif pour cet état.
  - *Etat 5* : Cet état est valide alors que son prédécesseur appartient à un meilleur chemin. La connaissance de validité admet donc un faux positif pour cet état.

- ***Connaissance d'optimalité des états*** :
  - *Etats 1 et 2* : ces deux états sont non optimaux et la connaissance d'optimalité les a bien désignés comme états non optimaux. La connaissance d'optimalité connaît donc deux vrais négatifs.
  - *Etats 3 et 4* : ces deux états sont optimaux et la connaissance d'optimalité les a bien désignés comme états optimaux. La connaissance d'optimalité connaît donc deux vrais positifs

- ***Connaissance de fin de cycle*** :
  - *Etats 1 et 2* : ces deux états appartiennent au meilleur chemin mais aucun d'eux n'est le meilleur état de ce meilleur chemin. La connaissance de fin de cycle connaît donc deux vrais négatifs.
  - *Etat 4* : Cet état est le meilleur état de l'arbre mais la connaissance de fin de cycle a laissé l'exploration se poursuivre. La connaissance de fin de cycle connaît donc un faux négatif.

E.IV.3 Analyse de l'historique

Dans le cadre de la généralisation automatique, il est préférable que les objets géographiques soient, dans leur ensemble, le mieux généralisés possible quitte à ce que quelques objets géographiques soient mal généralisés plutôt que de chercher à obtenir une généralisation de qualité homogène. Il est en effet préférable, pour les producteurs de données géographiques, de faire retoucher la généralisation de quelques objets géographiques à la main et d'obtenir un très bon résultat dans l'ensemble, plutôt que d'obtenir un résultat final de qualité moyenne sur l'ensemble des objets. Nous proposons donc, pour la note d'efficacité donnée par *l'agent diagnostic*, de prendre en compte la satisfaction moyenne obtenue sur l'échantillon de révision et de pondérer cette valeur par la valeur du premier quartile. Ainsi, une note élevée d'efficacité permettra d'assurer qu'au moins trois quarts des agents géographiques ont été bien généralisés.

Dans le cadre du modèle AGENT, la satisfaction des agents géographiques est comprise entre 1 et 10, il est donc possible de normer les satisfactions obtenues en les divisant par 10.

Nous notons respectivement *Moyenne(P_n)* et *Quartile(P_n)*, la moyenne et le premier quartile des satisfactions obtenues sur l'échantillon d'agents géographiques généralisés P_n.

L'agent diagnostic donnera la note d'efficacité suivante :

$$Note_{eff}(P_n) = \frac{Quartile(P_n) + Moyenne(P_n)}{20}$$

Concernant les notes données par les agents *connaissance*, nous proposons d'utiliser les fonctions suivantes : soit k, une connaissance et P_n, un échantillon d'instances de problème d'optimisation utilisé pour le diagnostic. Nous notons nb_{VP}, le nombre de vrais positifs, nb_{VN}, le nombre de vrais négatifs, nb_{FP}, le nombre de faux positifs et nb_{FN}, le nombre de faux négatifs.

- Pour les connaissances d'application des actions, de validité des états, et d'optimalité des états, nous proposons d'utiliser la fonction présentée en partie E.III.3. Nous calculons ainsi le taux de succès (nb_{VP} et nb_{VN}) par rapport à l'ensemble des cas rencontrés (nb_{VP}, nb_{VN}, nb_{FP} et nb_{FN}) :

$$Note(k, P_n) = \frac{nb_{VP} + nb_{VN}}{nb_{VP} + nb_{VN} + nb_{FP} + nb_{FN}}$$

- Pour les connaissances de fin de cycle, de restriction des actions et de priorité des contraintes : les vrais négatifs sont très nombreux par rapport aux vrais positifs, aux faux positifs et négatifs. Nous proposons donc de réduire leur influence en utilisant une racine carrée :

$$Note(k, P_n) = \frac{nb_{VP} + \sqrt{nb_{VN}}}{nb_{VP} + \sqrt{nb_{VN}} + nb_{FP} + nb_{FN}}$$

E.V Bilan

Nous avons proposé, dans ce chapitre, une approche de diagnostic en ligne de la qualité des connaissances du système de résolution de problèmes.

Notre approche est basée sur l'analyse des instances résolues du problème considéré. Cette analyse est assurée par les *agents connaissance* et par *l'agent diagnostic* et permet à ces derniers d'évaluer, sous forme de notes, la qualité du critère qu'ils représentent.

Nous avons proposé d'utiliser ces notes pour évaluer qualitativement, à l'aide d'une méthode de décision multicritère, la qualité globale du jeu de connaissances. Nous avons, à cet égard, présenté deux méthodes de décision multicritère que nous proposons d'utiliser dans le cadre de notre problème : la méthode ELECTRE TRI, fonctionnant par agrégation partielle des critères et une méthode basée sur la théorie des croyances, fonctionnant, elle, par agrégation complète des critères. Ces deux méthodes réputées et largement répandues sont particulièrement adaptées à notre problème dans lequel les critères sont peu fiables.

Nous avons enfin détaillé, dans une dernière partie, comment mettre en œuvre notre approche pour le modèle AGENT.

Nous avons présenté, au travers de ces cinq premiers chapitres, nos approches générales de révision et de diagnostic des connaissances procédurales d'un système fonctionnant par exploration informée d'arbres d'états. Nous proposons maintenant de présenter, dans un dernier chapitre, des expérimentations de ces approches menées dans le cadre d'une utilisation réelle pour la révision et le diagnostic des connaissances du modèle AGENT.

Chapitre F
Expérimentation et évaluation du modèle complet de révision des connaissances et du module de diagnostic

F.I Introduction

Nous avons proposé dans les chapitres précédents un ensemble de méthodes de révision des connaissances contenues dans des systèmes fonctionnant par exploration informée d'arbres d'états, ainsi qu'au diagnostic de la qualité de ces connaissances.

Notre approche générale de révision fonctionne hors ligne et consiste, dans une première phase *d'exploration*, à tracer le système de résolution de problèmes pendant la résolution d'un échantillon d'instances du problème considéré, puis dans une seconde phase *d'analyse*, à réviser le jeu de connaissances initial par analyse de ces traces (figure F.1).

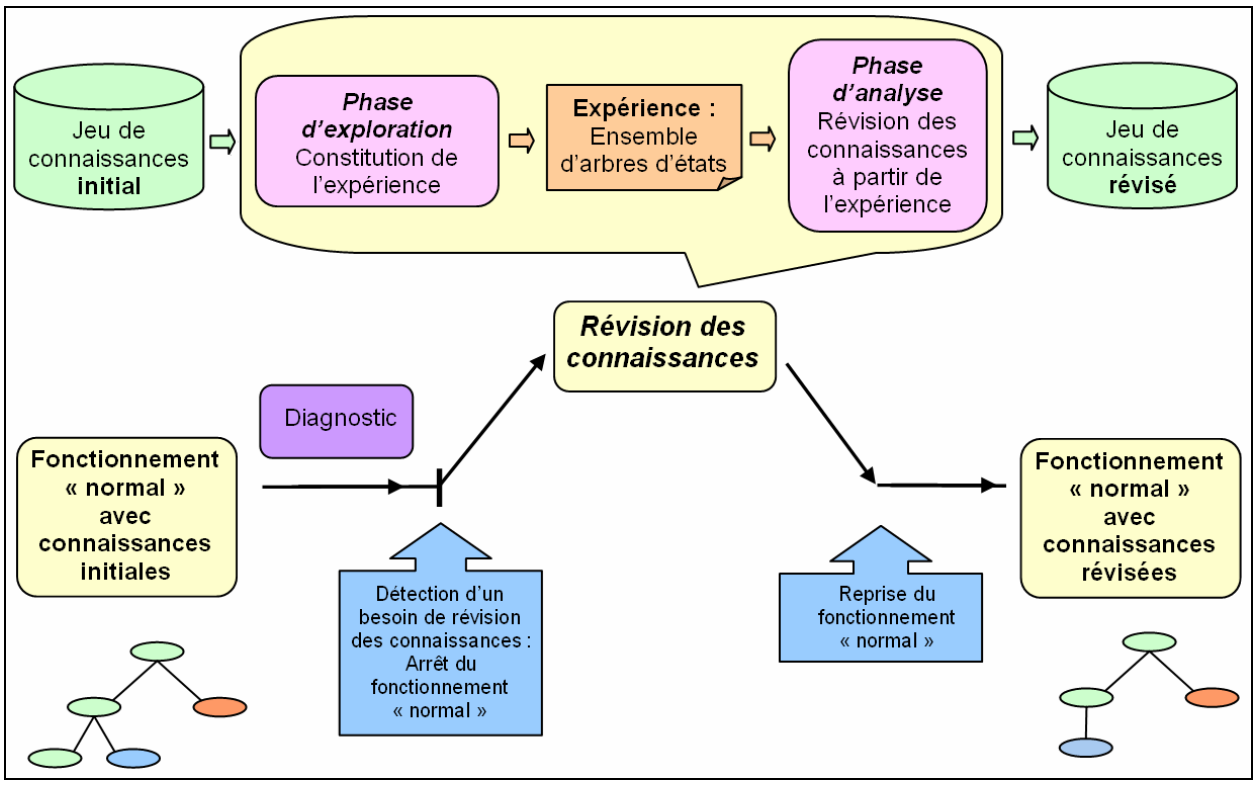


Figure F.1 Approche générale de révision des connaissances

Nous proposons dans ce dernier chapitre d'évaluer les différentes méthodes proposées au travers d'expérimentations menées avec le modèle AGENT.

Nous présentons en partie F.II le contexte général dans lequel ont été menées ces expérimentations.

La partie F.III est consacrée à la présentation des tests menés sur notre approche générale de révision et plus particulièrement sur les approches proposées pour la phase d'analyse (cf. D). Nous testons, en premier lieu, les approches destinées à la révision de chaque type de connaissances du modèle AGENT. Nous testons ensuite notre approche générale de révision de l'ensemble des connaissances. Ces expérimentations permettent également d'analyser l'influence des différents paramètres que nos méthodes demandent de définir.

Nous proposons ensuite en partie F.IV d'étudier l'influence du choix de l'échantillon de révision dans le processus de révision globale. Nous analysons pour cela les résultats de révision obtenus à partir de différents échantillons de révision.

Nous nous intéresserons en partie F.V à l'expérimentation de notre approche d'évaluation de jeux de mesures. Nous proposons dans ce cadre d'évaluer un ensemble de jeux de mesures et d'étudier l'adéquation entre la qualité des connaissances pouvant être obtenues à partir de ces jeux et l'évaluation donnée par notre approche.

Nous proposons enfin en partie F.VI une expérimentation de notre approche de diagnostic de la qualité de connaissances.

F.II Contexte général des expérimentations

F.II.1 Données et types d'objets géographiques considérés : les groupements de bâtiments

F.II.1.1 Introduction

Nous présentons dans cette partie F.II.1, la classe d'agents géographiques *groupement de bâtiments*. Toutes nos expérimentations sont basées sur la généralisation cartographique au 1 : 50 000 de ces agents meso à partir de données provenant de la BDTopo® de l'IGN dont l'échelle de référence est 1 : 15 000.

F.II.1.2 Caractéristiques des agents *groupement de bâtiments*

La généralisation des espaces urbains est un problème particulièrement complexe qui demande de traiter de fortes densités d'information. Il est souvent nécessaire, afin de généraliser ces espaces, de recourir à différents niveaux d'analyses (cf. A.IV.1). Effectivement, certains conflits cartographiques doivent être résolus au niveau d'objets pris individuellement alors que d'autres doivent être résolus au niveau de groupes d'objets. Différents travaux se sont intéressés à la définition de ces niveaux d'analyse (Ruas, 1999 ; Boffet, 2000 ; Steiniger et al., 2008).

Nous proposons dans notre cadre d'utiliser comme niveau d'analyse les *groupements de bâtiments* définis par (Regnauld, 1998 ; Bard, 2004 ; Gaffuri & Trévisan, 2004). Un *groupement de bâtiments* représente un ensemble de bâtiments « *proches* » d'un même îlot.

Le choix de cette classe d'agents géographiques pour nos expérimentations n'est pas anodin. En effet, ces agents sont particulièrement intéressants car il est difficile de définir de bonnes connaissances pour leur généralisation. Contrairement aux bâtiments pris individuellement pour lesquels la généralisation au 1 : 50 000 est bien maîtrisée, celle des *groupements de bâtiments* pour cette même échelle pose de nombreux problèmes d'efficience et d'efficacité.
Les problèmes sont d'autant plus importants que les actions nécessaires à leur généralisation sont pour la plupart assez lourdes d'un point de vue temps et ressources de calcul. Il est donc indispensable de réussir à trouver un état acceptable en parcourant le moins d'états possibles, et définir des connaissances permettant cela s'avère extrêmement complexe.

Ainsi, à l'heure actuelle, en raison de l'incapacité à définir des connaissances suffisamment robustes pour assurer leur généralisation, la généralisation des *groupements de bâtiments* est assurée, dans le cadre de la production de cartes à l'IGN au 1 : 50 000, par application d'une même séquence fixe d'actions pour tous les *groupements de bâtiments*. Disposer de bonnes connaissances permettrait, en appliquant un nombre raisonnable d'actions, d'obtenir des résultats de généralisation bien meilleurs.

La création de ces *groupements de bâtiments* est issue d'un processus comprenant trois étapes dont les deux premières sont issues de (Boffet, 2001). La première étape consiste à créer des objets *villes* par regroupement de bâtiments dont la proximité est inférieure à un seuil défini. La seconde étape consiste à découper chaque ville en îlots urbains. Un îlot est un ensemble de

bâtiments entouré de tronçons routiers qui bouclent minimalement. La dernière étape consiste à construire, au sein de chaque îlot, des *groupements de bâtiments* par regroupement des bâtiments dont la proximité est inférieure à un seuil défini.

(Boffet, 2001) a proposé de caractériser ces *groupements de bâtiments* en se basant sur leur densité en bâtiments ainsi que sur les caractéristiques des bâtiments qui les composent. Nous utiliserons cette caractérisation pour différencier les groupements de type « *centre-ville* » des autres *groupements de bâtiments*. Effectivement, les *groupements de bâtiments* de ce type seront généralement généralisés de façons très différentes des *groupements de bâtiments* des autres types. Ainsi, pour une généralisation au 1 : 50 000, les *groupements de bâtiments* de type *centre-ville* sont souvent directement remplacés par une surface. La figure F.2, qui présente un extrait de carte au 1 : 50 000, illustre ces propos : sur cet extrait, les groupements situés en centre-ville ont directement été remplacés par une surface grise contrairement aux autres groupements de bâtiments.

Figure F.2 Extrait de carte au 1 : 50 000

Le modèle AGENT permet de gérer différents niveaux d'analyse au travers des agents micro et meso. Nous rappelons qu'un agent micro représente un objet géographique pris individuellement, alors qu'un agent micro représente un groupe d'objets géographiques (cf. A.III.1).

Un agent *groupement de bâtiments* est un agent meso qui délimite l'emprise spatiale du *groupement de bâtiments*. Il est composé d'au moins un agent micro *bâtiment* et potentiellement d'agents micro d'autres classes tels que des agents *tronçon de route* ou des agents *tronçon de rivière*.

Concernant la généralisation des agents micro composant l'agent méso, nous nous intéresserons uniquement à la généralisation des agents *bâtiments* et non à celle des autres agents géographiques. En effet, en raison de la complexité de la généralisation au 1 : 50 000 des espaces urbains, la généralisation des autres agents micro (en particulier celle des agents *tronçon de route* ou des agents *tronçon de rivière*) est souvent effectuée préalablement à la généralisation des *groupements de bâtiment*.

F.II.1.3 Contraintes utilisées

Nous avons utilisées six contraintes pour les agents *groupement de bâtiments* (figure F.3). Nous nous sommes inspiré des travaux de (Ruas, 1999) pour la définition de ces contraintes.

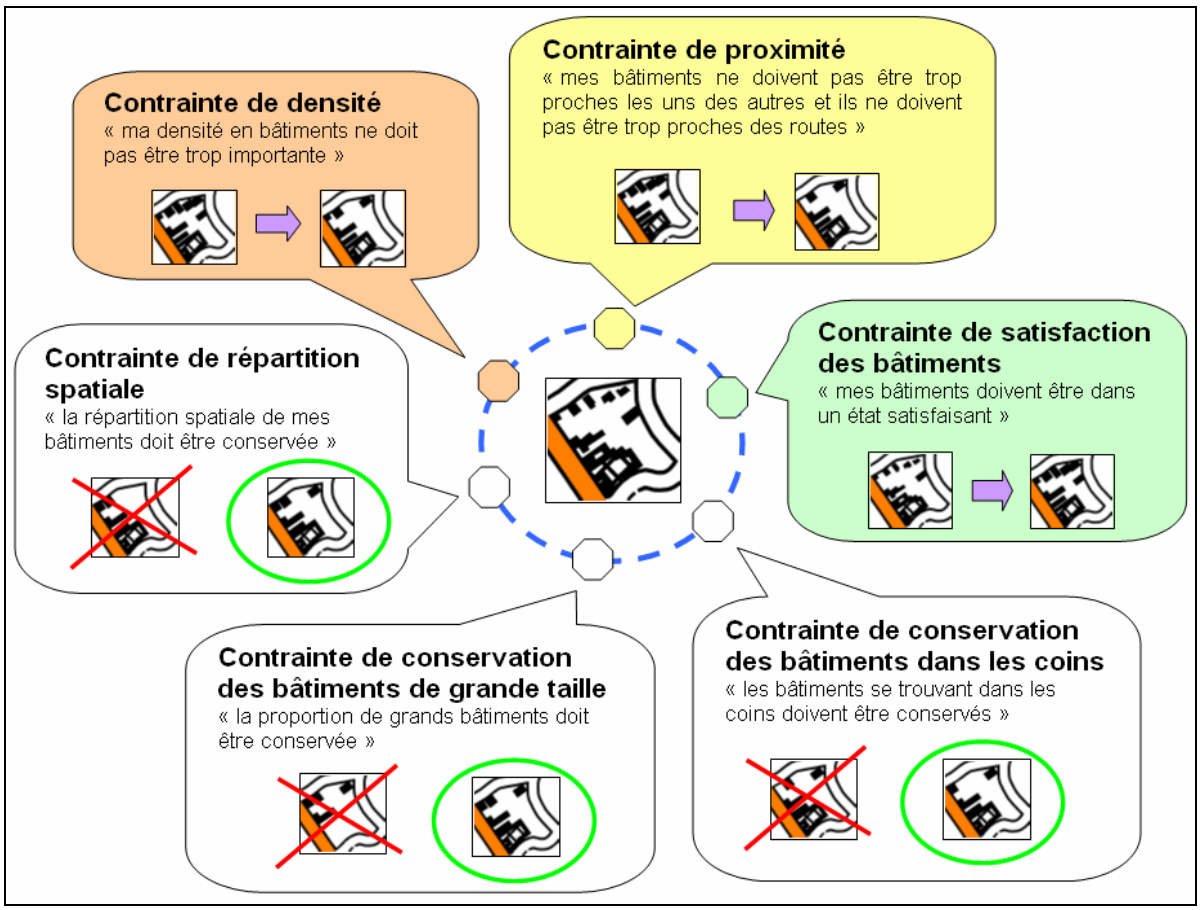


Figure F.3 Contraintes des agents *groupement de bâtiments*

Ces contraintes sont :

- ***La contrainte de proximité*** : cette contrainte a pour rôle d'assurer que les bâtiments ne soient pas trop proches les uns des autres et qu'ils ne soient pas trop proches des routes. Elle propose à l'agent géographique des actions visant à déplacer des bâtiments ou à en supprimer.
- ***La contrainte de satisfaction des bâtiments*** : cette contrainte assure que les bâtiments composant le groupement de bâtiments sont bien généralisés (satisfaits de leur état interne). Elle propose à l'agent géographique de déclencher la généralisation des agents *bâtiments*.
- ***La contrainte de densité*** : cette contrainte a pour rôle de vérifier que la densité en bâtiments courante n'est pas trop élevée par rapport à la densité initiale. Elle propose à l'agent géographique de supprimer des bâtiments
- ***La contrainte de répartition spatiale*** : cette contrainte a pour rôle de vérifier que la répartition spatiale courante des bâtiments est proche de leur répartition initiale, c'est à dire qu'aucun espace vide n'est apparu là où il y avait des bâtiments à l'état initial. Cette contrainte ne propose pas d'action.

- ***La contrainte de conservation des bâtiments de grande taille*** : cette contrainte a pour rôle de vérifier que les bâtiments de grande taille ont été conservés. Elle ne propose pas d'action.
- ***La contrainte de conservation des bâtiments dans les coins*** : cette contrainte a pour rôle de vérifier que les bâtiments se trouvant dans les « *coins* » de l'îlot n'ont pas été supprimés. En effet, les bâtiments se trouvant au niveau d'un angle droit formé par des tronçons routiers servent souvent de marqueur visuel, il est donc intéressant de les conserver lors de la généralisation. Nous présentons en annexe *An.1.2* l'algorithme que nous avons développé pour la détection des bâtiments se trouvant dans les « coins ». Cette contrainte ne propose pas d'action.

F.II.1.4 Description des actions

Nous avons retenu six actions possibles pour les agents *groupement de bâtiments* (figure F.4). Chacune de ces actions est liée à une contrainte.

- Contrainte de satisfaction des bâtiments :
  - ***Action de généralisation des bâtiments*** : cette action active la généralisation des agents micro *bâtiment* composant l'agent meso *groupement de bâtiments*.

- Contrainte de proximité :
  - ***Action de déplacement de bâtiments*** : cette action déplace les bâtiments souffrant d'un problème de proximité. Cette action est une réimplémentation partielle de l'algorithme de déplacement proposé par (Ruas, 1999, p.177).
  - ***Action de suppression locale de bâtiments*** : cette action supprime les bâtiments en fonction d'un contexte local. Elle supprime en priorité les bâtiments qui ont le plus de problèmes de superposition avec les autres bâtiments. Cet algorithme, développé spécialement pour ces expérimentations, est présenté en annexe *An.1.3*.
  - ***Action de suppression/recentrage d'un bâtiment*** : cette action sélectionne le bâtiment qui pose le plus de problème de proximité et le supprime. Si un autre bâtiment se trouve à proximité, ce dernier est déplacé de manière à se rapprocher du bâtiment supprimé. Cette action est une réimplémentation de la dernière étape effectuée par l'algorithme de déplacement proposé par (Ruas 1999, p.186).

- Contrainte de densité :
  - ***Action de suppression globale de bâtiments*** : cette action supprime les bâtiments en fonction du contexte global. Elle supprime en priorité les bâtiments les plus congestionnés (ceux qui ont le moins de possibilité pour se déplacer). Cette action est une réimplémentation de l'algorithme proposé par (Ruas 1999, p.171).

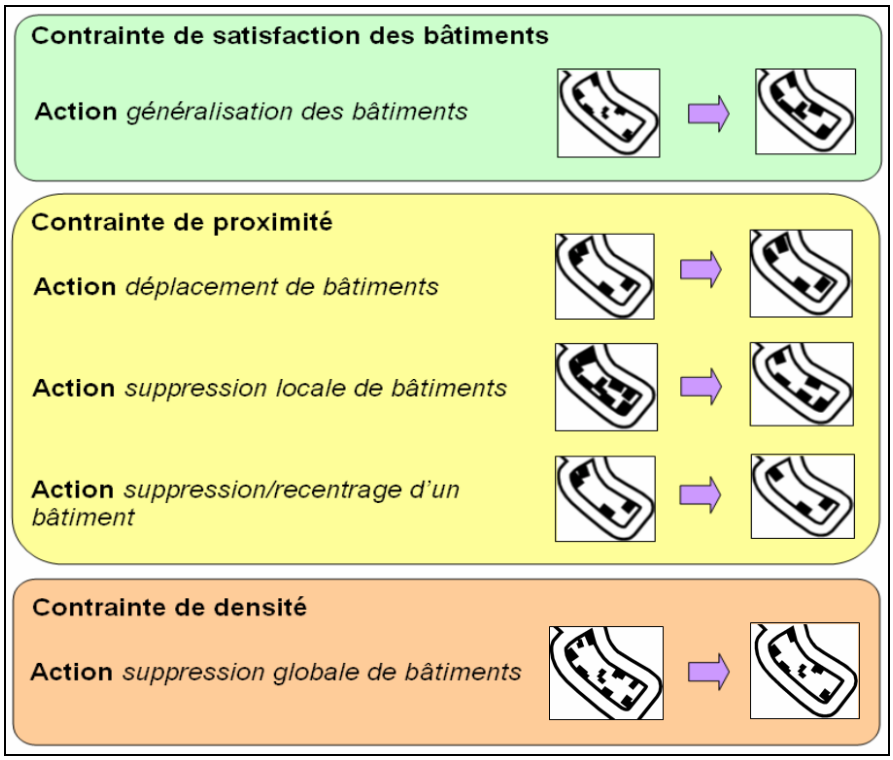


Figure F.4 Action des agents *groupement de bâtiments*

F.II.1.5 Description des jeux de mesures

Nous avons défini pour les agents *groupement de bâtiments* un ensemble de mesures décrivant leur état géométrique :

- *mediane_satisfaction_bâti* : médiane des satisfactions des agents *bâtiment* composant le *groupement de bâtiments*.
- *prem_quartileSatisfaction_bâti* : premier quartile des satisfactions des agents *bâtiment* composant le *groupement de bâtiments*.
- *min_satisfaction_bâti* : satisfaction minimale des satisfactions des agents *bâtiment* composant le *groupement de bâtiments*.
- *moyenne_proximité* : moyenne des proximités entre bâtiments et entre bâtiments et routes. La proximité entre deux objets géographiques correspond à la distance minimale entre eux, c'est-à-dire calculée aux points les plus proches. Cette proximité est calculée en mètres. Elle peut être négative dans le cas où les deux objets sont superposés et correspond alors à la distance maximale de superposition.
- *médiane_proximité* : médiane des proximités entre bâtiments et entre bâtiments et routes.
- *prem_quartile_proximité* : premier quartile des proximités entre bâtiments et entre bâtiments et routes.
- *min_proximité* : valeur minimale des proximités entre bâtiments et entre bâtiments et routes.
- *ratio_surface_superposition* : taux de surfaces de bâtiments superposées à d'autres bâtiments ou à des routes par rapport à la surface totale des bâtiments.
- *nb_superpositions* : nombre de bâtiments superposés à d'autres bâtiments ou à des routes.
- *nb_bâtiments* : nombre de bâtiments.

- *nb_grands_bâtiments* : nombre de bâtiments de grande taille. Un bâtiment est considéré comme de grande taille si sa surface dépasse un certain seuil.
- *aire* : aire du *groupement de bâtiments*.
- *densité* : densité en termes de surface de bâtiments par rapport à l'aire du *groupement de bâtiments*.
- *nb_coins_bâtiments* : nombre de « *coins* » de l'îlot occupés par un ou plusieurs bâtiments.

Nous prendrons en compte dans la définition des différentes connaissances procédurales, en plus de ces mesures, les satisfactions des contraintes.

Nous rappelons que pour des raisons de formalisation, nous avons proposé dans notre version enrichie du modèle AGENT de définir un jeu de mesures pour chaque contrainte ainsi qu'un jeu de mesures pour l'agent géographique (cf. B.V.3.1).

Nous proposons pour le jeu de mesures lié à l'agent géographique de prendre en compte l'ensemble des mesures définies. Concernant les contraintes, nous proposons pour chaque contrainte de prendre en compte l'ensemble des mesures utilisées par celle-ci dans le calcul de sa satisfaction. En effet, ces mesures ont un lien direct avec la contrainte et ont donc plus de chance de caractériser pertinemment le critère que la contrainte représente.

Les jeux de mesures utilisés sont donc :
- pour la contrainte de *satisfaction des bâtiments* : {$satisfaction_{satisfactionBât}$, *mediane_satisfaction_bâti*, *prem_quartileSatisfaction_bâti*, *min_satisfaction_bâti*}
- pour la contrainte de *proximité* : {$satisfaction_{proximit,}$ *moyenne_proximité*, *médiane_proximité*, *prem_quartile_proximité*, *min_proximité*, *ratio_surface_superposition*, *nb_superpositions*}
- pour la contrainte de *densité* : {$satisfaction_{densité}$, *nb_bâtiments*, *nb_bâtiments_importants*, *aire*, *densité*}
- pour la contrainte de *répartition spatiale* : {$satisfaction_{repartitionSpat}$}
- pour la contrainte de *conservation des bâtiments importants* : {$satisfaction_{bâtimentsImp}$, *nb_bâtiments*, *nb_bâtiments_importants*}
- Jeu de mesures lié à la contrainte de *conservation des bâtiments dans les coins* : {$satisfaction_{bâtimentscoins}$}
- pour les agents *groupements de bâtiments* : ensemble des mesures définies

La figure F.5 donne des exemples de valeurs de ces mesures pour différents *groupements de bâtiments* dans leur état initial. A noter que sur cet exemple nous n'avons pas donné de valeurs pour la satisfaction de toutes les contraintes. En effet, la satisfaction de certaines contraintes ne peut être calculée que dans le cadre d'une généralisation car le calcul de celle-ci est basé sur la comparaison de l'état courant de l'agent géographique avec son état initial.

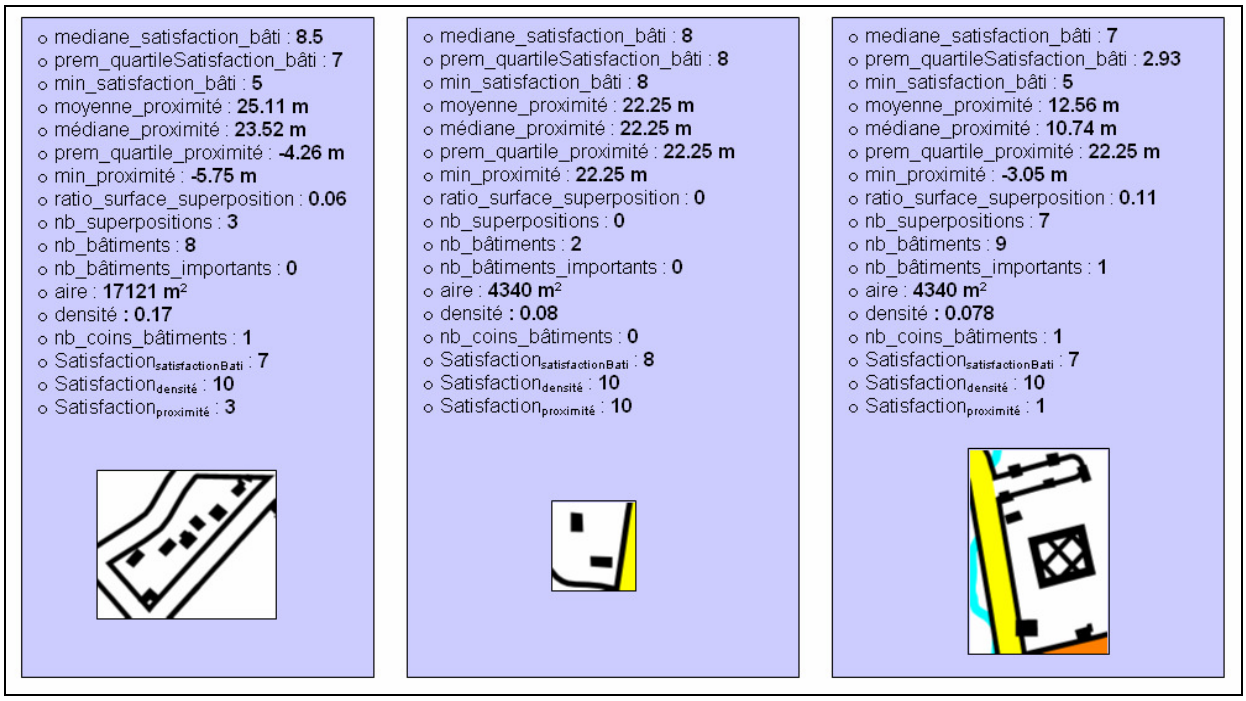


Figure F.5 Exemples de valeurs de mesures pour des *groupements de bâtiments*

F.II.2 Zone de révision des connaissances et zone de test

Nous proposons de définir deux zones géographiques distinctes. La première est destinée à la mise au point de jeux de connaissances ainsi qu'à leur révision. La seconde est destinée à l'évaluation des jeux de connaissances. Il est important de disposer de deux zones indépendantes pour la révision et les tests afin de garantir que le processus de révision permet de fournir un jeu de connaissances performant pour l'ensemble des *groupements de bâtiments* et pas uniquement pour les *groupements de bâtiments* d'une zone particulière (celle qui a servi d'échantillon pour la révision). Cette précaution permettra de s'assurer que le processus de révision ne souffre pas de surapprentissage.

Nous avons utilisé comme zone de révision la ville d'Orthez. Cette ville Française est composée d'environ 280 *groupements de bâtiments*. La figure F.6 présente cette ville (zone située à l'intérieur de la délimitation orange) tirée de la BDTopo® de l'IGN avec et sans symbolisation au 1 : 50 000. Nous avons utilisé 50 *groupements de bâtiments* de cette zone pour réviser les connaissances. Le nombre 50 a été défini empiriquement en prenant en compte le nombre total de *groupements de bâtiments* disponibles (280) et du temps moyen nécessaire à la généralisation d'un *groupement de bâtiments* avec un jeu de connaissances *minimal*. Ces 50 groupements de bâtiments ont été sélectionnés automatiquement par notre méthode de choix d'un échantillon représentatif (cf. C.II.2). Nous revenons plus en détail sur l'influence de ce choix en partie F.IV.

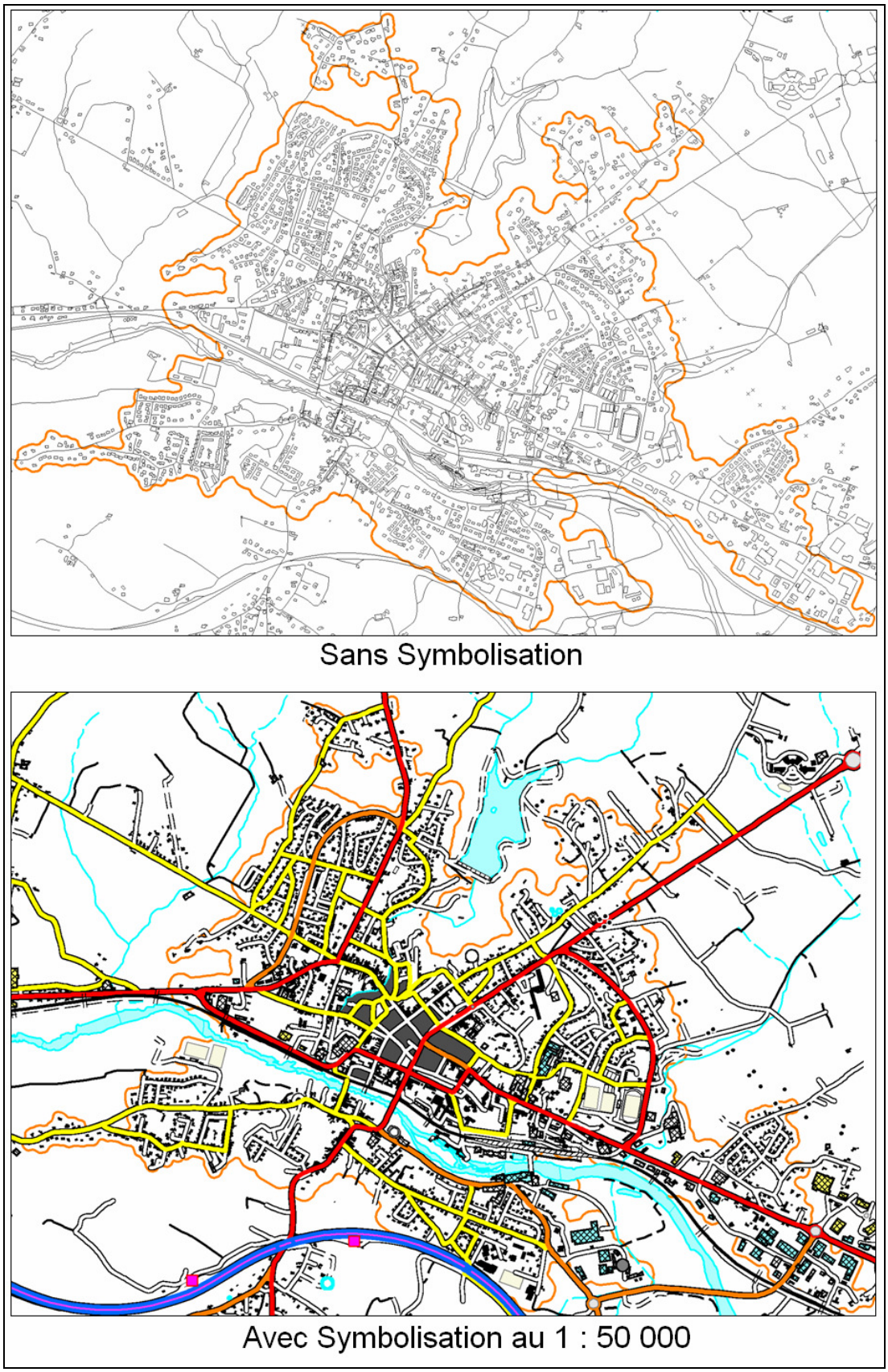


Figure F.6 Zone de révision : ville d'Orthez

Nous avons utilisé comme zone de test la ville de Salies-de-Béarn. Cette ville Française est composée d'environ 200 *groupements de bâtiments*. La figure F.7 présente cette ville (zone située à l'intérieur de la délimitation orange) tirée de la BDTopo® de l'IGN avec et sans symbolisation au 1 : 50 000. Les 200 *groupements de bâtiments* de cette zone seront utilisés pour les tests.

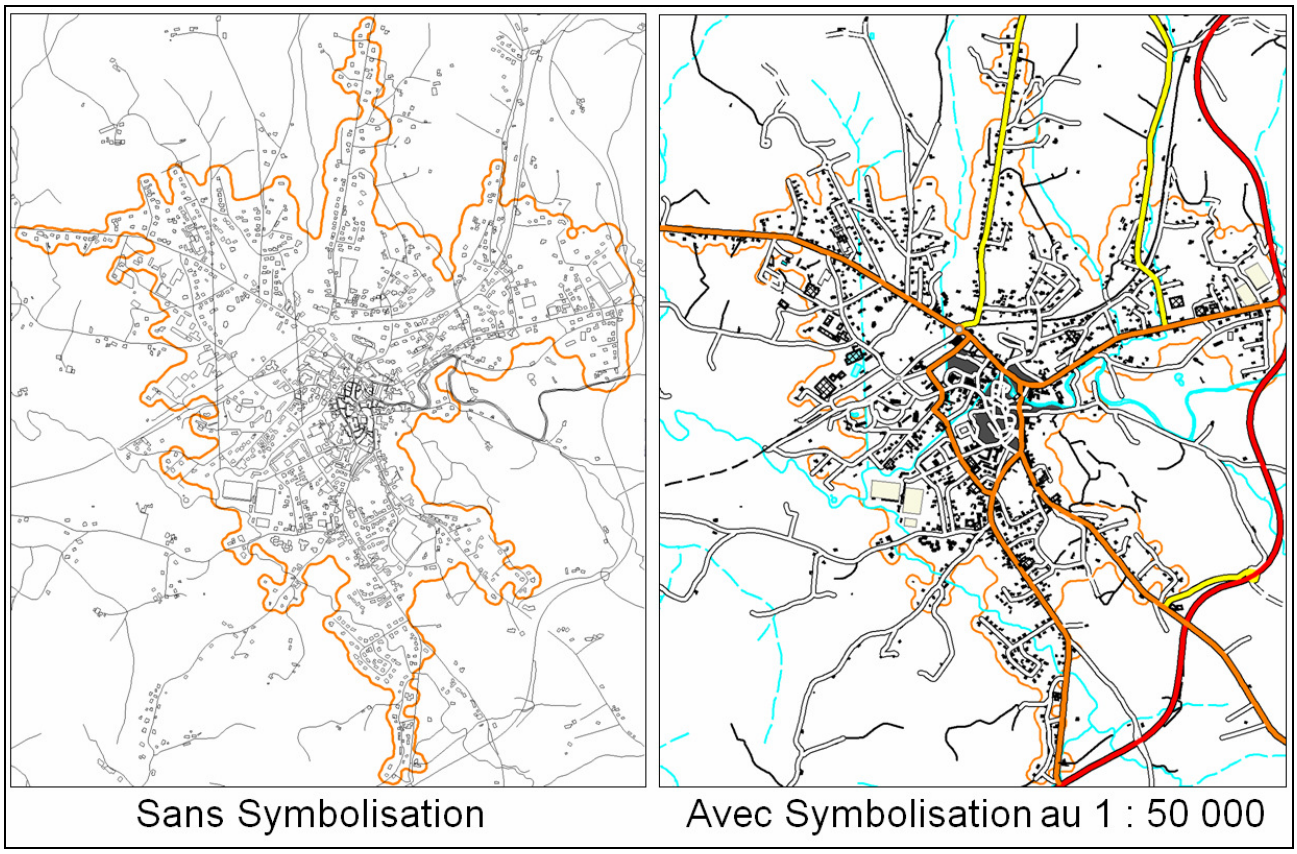


Figure F.7 Zone de test : ville de Salies-de-Béarn

F.II.3 Jeux de connaissances initiaux définis

L'objectif de la révision des connaissances est d'améliorer un jeu de connaissances initial grâce à l'apport de nouvelles informations. Une question importante dans le cadre de la validation de notre approche de révision concerne les jeux de connaissances initiaux utilisés dans les expérimentations.

Nous proposons donc de définir trois jeux de connaissances : le premier est un jeu dit « *basique* » qui correspond à une absence complète de connaissances spécifiques à la généralisation des *groupements de bâtiments*, le second jeu a été défini par un expert en cartographie qui ne maîtrise pas le modèle AGENT, le dernier jeu a été défini par un expert en cartographie qui maîtrise le modèle AGENT[1]. Afin de distinguer les deux jeux de connaissances définis par les experts, nous noterons « *jeu de connaissances défini par l'expert en cartographie* » le jeu défini par l'expert ne maîtrisant pas le modèle AGENT et par « *jeu de connaissances défini par l'expert du modèle AGENT* » celui défini par l'expert maîtrisant le modèle AGENT.

[1] Nous remercions Laurence Jolivet et Julien Gaffuri pour leur participation à ce test

La figure F.8 présente le jeu de connaissances *basique*. Comme illustré sur cette figure, ce jeu de connaissances correspond en tout point à un jeu de connaissances *minimal* (cf. C.III.2) à l'exception du fait que l'agent géographique se désactivera pour notre jeu *basique* lorsqu'un état parfait sera atteint. Ce jeu de connaissances permet donc de trouver, pour chaque agent géographique généralisé, le meilleur état possible en termes de satisfaction compte tenu des contraintes et des actions définies. Ce jeu servira donc de référence en termes de qualité des résultats obtenus. Dans ce jeu de connaissances, toutes les contraintes ont une priorité de 1 pour tous les états. Les actions, proposées pour tous les états, ont toutes un poids de 1. Les restrictions sur les actions sont toutes égales au nombre maximal d'utilisations des actions ACT_MAX (cf. B.V.3.1.2) auquel nous avons donné empiriquement une valeur de 10 pour les *groupements de bâtiments*. Aucune règle de fin de cycle, ni d'optimalité des états n'est définie. Le critère de validité sélectionné est le critère ***CEPAC*** qui est utilisé par les jeux de connaissances *minimaux* : un état est valide si la satisfaction de la contrainte qui a proposé l'action ayant mené à cet état a augmenté (cf. B.V.3.1.2).

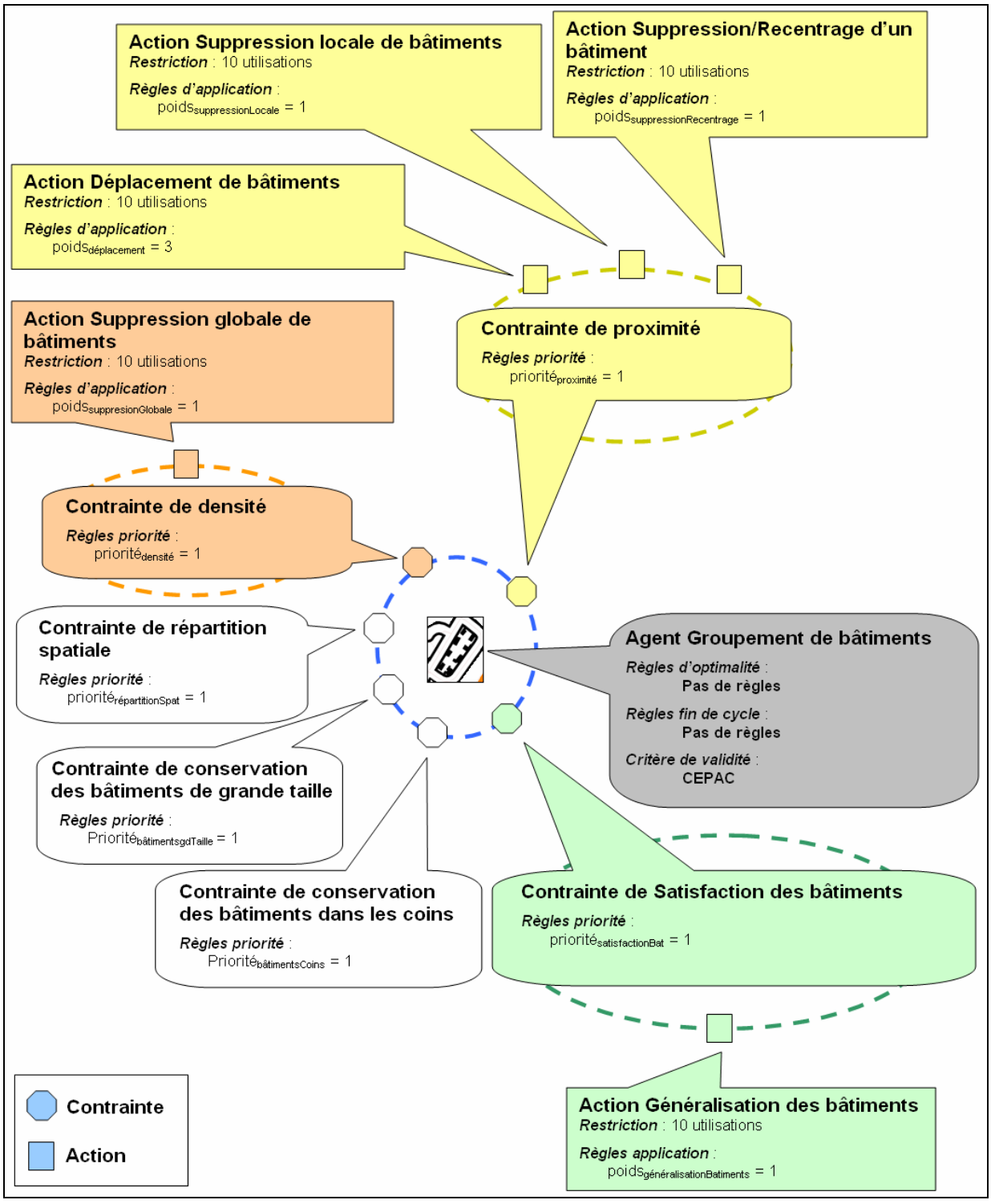


Figure F.8 Jeu de connaissances *basique*

Les deux autres jeux de connaissances ont été définis par des experts. Ces derniers ont défini leur jeu de connaissances à partir de tests et d'analyses menés sur les *groupements de bâtiments* de la zone de révision.

Nous avons fournis aux experts les outils suivants pour les aider à définir leur jeu de connaissances :

- Une interface leur permettant de tester chacune des actions séparément sur un agent *groupement de bâtiments*.
- Une interface leur permettant de connaître les valeurs de chaque mesure pour un agent *groupement de bâtiments*.
- La possibilité de modifier directement un jeu de connaissances et de le charger dans le système de généralisation.
- La possibilité d'activer la généralisation d'un agent *groupement de bâtiments* (avec le jeu de connaissances chargé)
- Une interface permettant de visualiser l'arbre d'états construit lors de la généralisation d'un agent *groupement de bâtiments* ainsi que la géométrie de ses sous-agents et les valeurs de l'ensemble des mesures pour chacun des états visités.

Une fois les jeux de connaissances définis, nous avons interviewé les experts afin de connaître la démarche qu'ils ont utilisée pour définir leur jeu de connaissances.

La figure F.9 détaille le jeu de connaissances défini par l'expert en cartographie. La démarche de l'expert en cartographie dans sa définition de son jeu de connaissances a d'abord été de chercher à définir un jeu de connaissances efficace qui fournit des résultats cartographiques de bonne qualité. Il a ensuite cherché à améliorer l'efficience du processus en remplaçant le critère de validité initialement choisi (***CEPAC***) par un critère permettant un élagage plus sévère (***CEPAC ET CEEAP***) ainsi que par l'introduction de règles d'optimalité des états et de fin de cycle. Concernant l'ordre de traitement des conflits (la priorité des contraintes), il a proposé, lorsque l'agent *groupement de bâtiments* souffre de graves problèmes de proximité, de satisfaction des sous-agents et de densité, de régler en priorité les problèmes de proximité, puis ceux de satisfaction des bâtiments, et enfin ceux de densité. Pour les connaissances d'application des actions, l'expert a proposé d'appliquer toutes les actions pour tous les états sauf les deux actions de suppression de bâtiments proposées par la contrainte de proximité qui ne sont appliquées que lorsque la contrainte de proximité est très insatisfaite (satisfaction < 6). Concernant les actions proposées par la contrainte de proximité, l'expert a proposé d'appliquer en priorité l'action de déplacement, puis l'action de suppression/recentrage d'un bâtiment et enfin l'action de suppression locale de bâtiments.

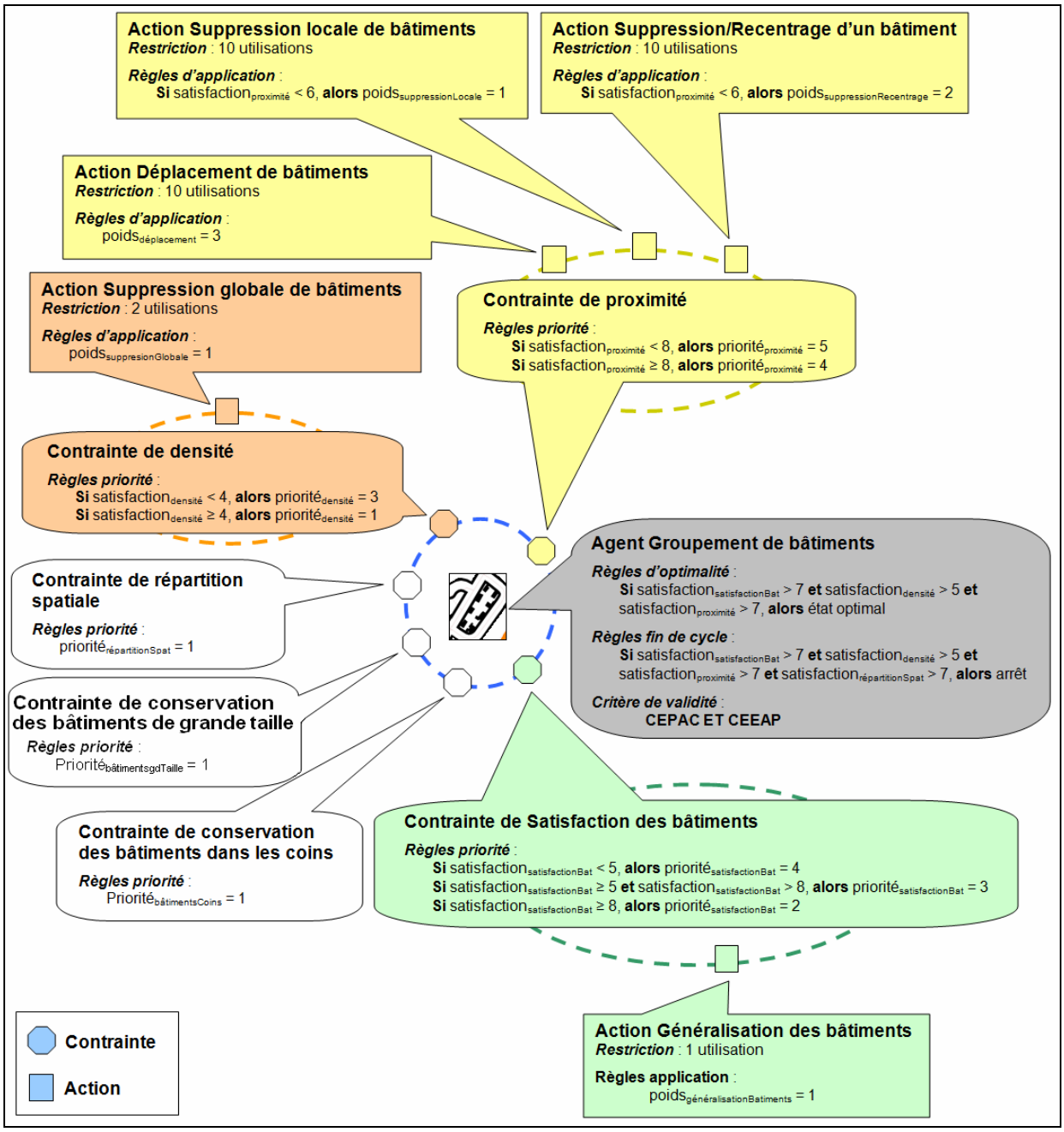


Figure F.9 Jeu de connaissances défini par l'expert en cartographie

La figure F.10 détaille le jeu de connaissances défini par l'expert du modèle AGENT. La démarche de l'expert du modèle AGENT dans sa définition de son jeu de connaissances a été similaire à celle de l'expert en cartographie : il a en premier lieu cherché à définir un jeu de connaissances efficace puis a ensuite cherché à améliorer son efficience. Un point important est que l'expert du modèle AGENT a volontairement choisi de favoriser l'efficacité par rapport à l'efficience.

L'expert du modèle AGENT a d'abord défini les règles de priorité des contraintes et d'application des actions. Il a ensuite choisi le même critère de validité que celui choisi par l'expert en cartographie (***CEPAC ET CEEAP***) puis a ajouté des restrictions sur le nombre d'utilisations de certaines actions. L'expert du modèle AGENT n'a par contre pas défini de règles d'optimalité ni de fin de cycle. Concernant l'ordre de traitement des conflits, il a

proposé, lorsque l'agent *groupement de bâtiments* souffre de problèmes de densité, de satisfaction des sous-agents et de proximité, de traiter en priorité les problèmes de densité puis ceux de proximité et enfin ceux de satisfaction des bâtiments. Pour les connaissances d'application des actions, l'expert a proposé d'appliquer toutes les actions pour tous les états sauf l'action de suppression de bâtiments proposée par la contrainte de densité qui n'est appliquée que lorsque le *groupement de bâtiments* est composé d'au moins deux bâtiments. Concernant les actions proposées par la contrainte de proximité, l'expert a proposé d'appliquer en priorité lorsque la contrainte est très insatisfaite l'action de suppression/recentrage d'un bâtiment, puis celle de suppression locale de bâtiments et enfin celle de déplacement. Inversement, lorsque la contrainte de proximité est moins insatisfaite, l'action appliquée en priorité est celle de déplacement, puis celle de suppression/recentrage d'un bâtiment et enfin celle de suppression locale de bâtiments.

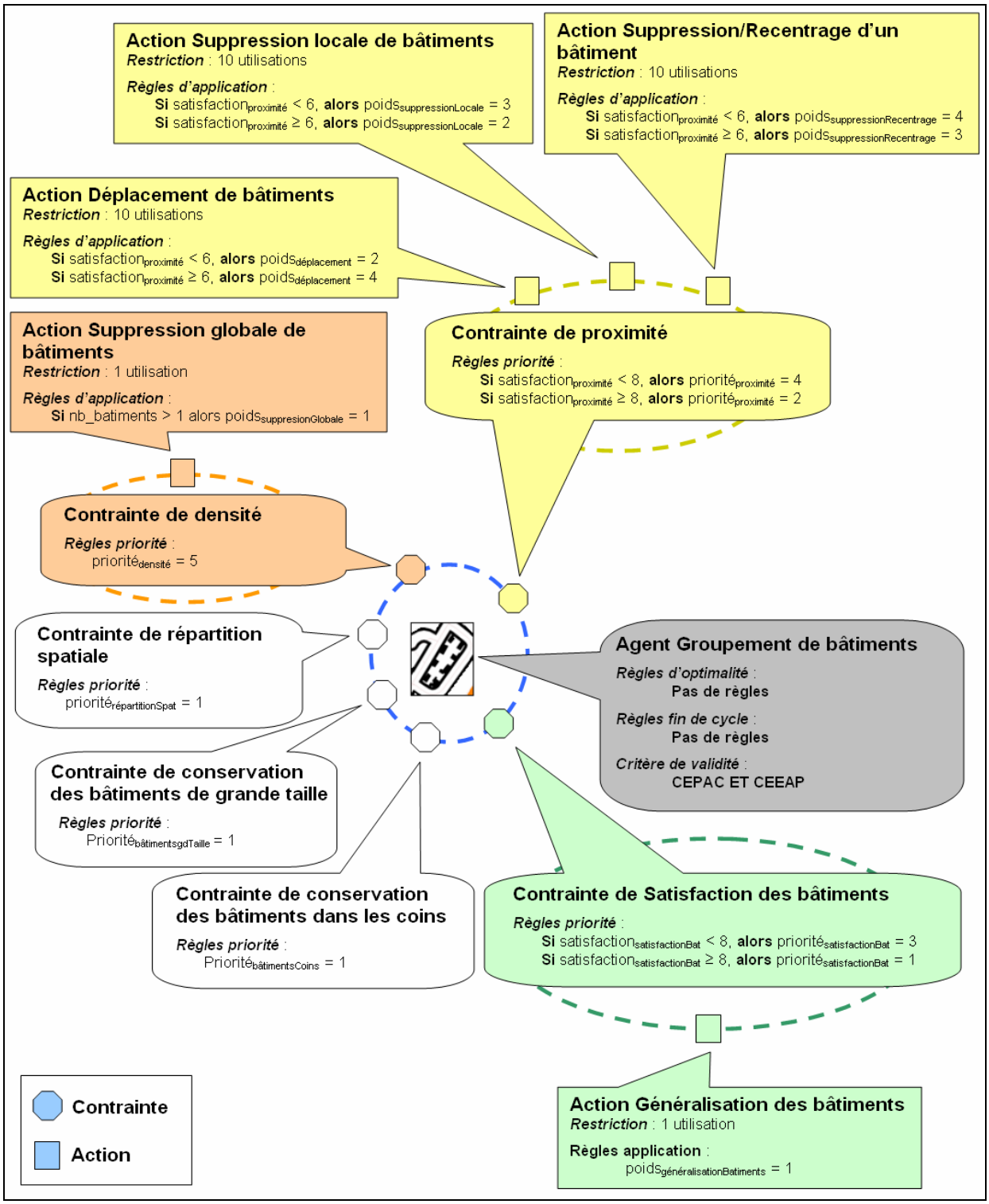


Figure F.10 Jeu de connaissances défini par l'expert du modèle AGENT

Nous avons testé nos trois jeux de connaissances sur la zone de test. Les figures F.11 et F.12 donnent les résultats obtenus pour chaque *groupement de bâtiments* en termes de satisfaction (en ordonnée) et de nombre d'états parcourus (en abscisse). Idéalement, le résultat d'une généralisation devrait se trouver en haut (la meilleure satisfaction possible) à gauche (le moins d'états parcourus possible). Comme nous pouvons le constater sur la figure F.10, le jeu de connaissances *défini par l'expert du modèle AGENT* a permis d'obtenir des résultats

légèrement meilleurs en termes de satisfaction que le jeu de connaissances *défini par l'expert en cartographie* (plus de *groupements de bâtiments* ont obtenu une satisfaction de 10 ou proche de 10). Cependant, le jeu de connaissances *défini par l'expert en cartographie* a permis au système de généralisation d'être plus efficient. On peut également remarquer que quatre agents *groupement de bâtiments* n'ont pas, à tort, été généralisés avec le jeu de connaissance *défini par l'expert en cartographie* (les quatre points qui se trouvent sur l'axe « nombre d'états parcourus = 1 »). Cette absence de généralisation est due à la règle de fin de cycle définie par l'expert en cartographie.

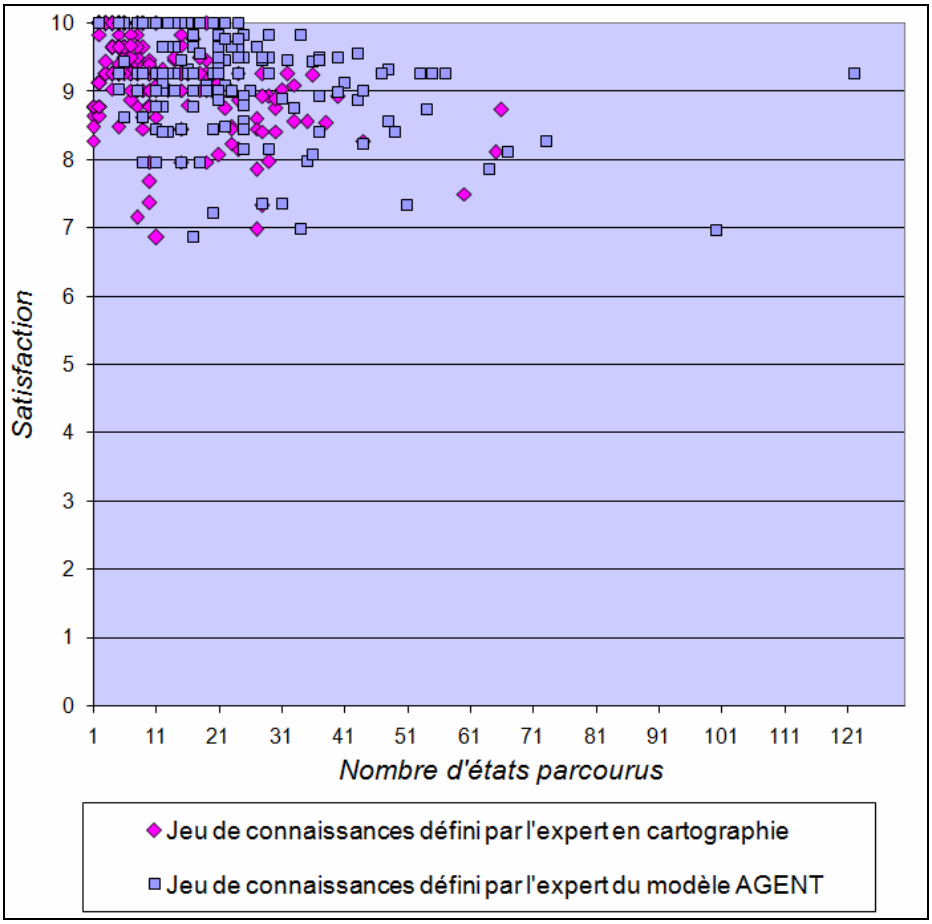


Figure F.11 Résultats de généralisation obtenus avec les jeux de connaissances définis par les experts sur la zone de test

Concernant les performances obtenues par le jeu de connaissances *basique* (figure F.12), si celui-ci a bien permis d'obtenir les meilleurs résultats en termes de satisfaction, le nombre d'états nécessaires à la généralisation des agents *groupement de bâtiments* rend son utilisation totalement inadaptée à une application réelle.

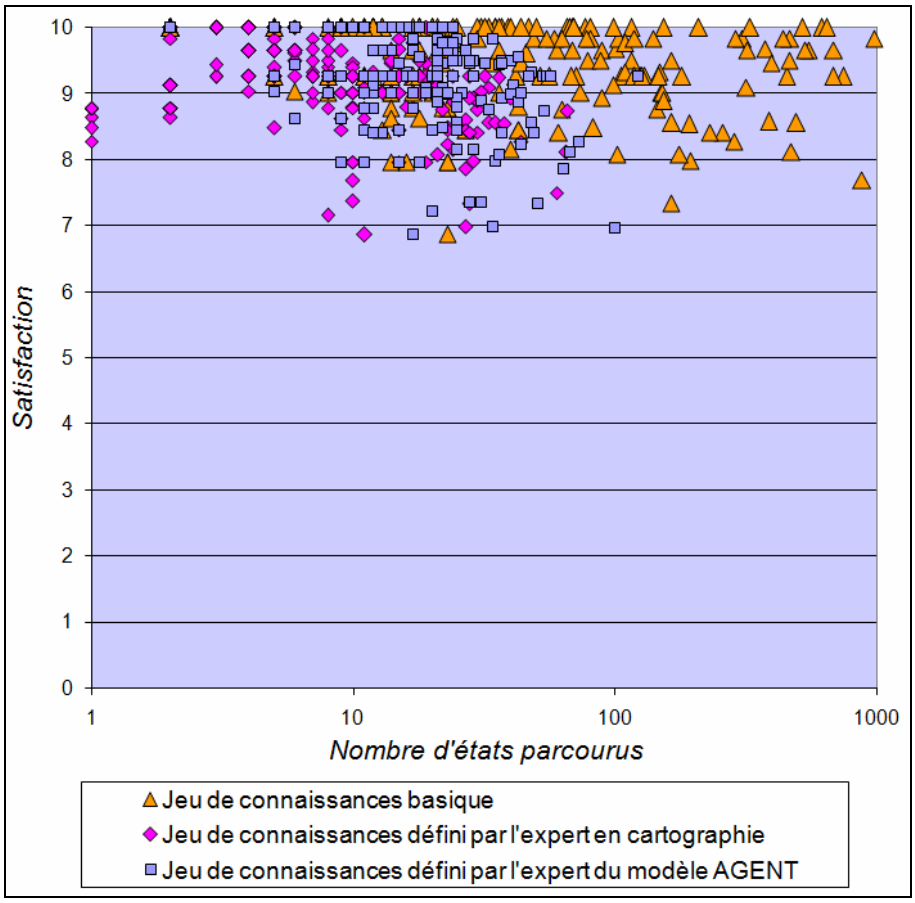


Figure F.12 Résultats de généralisation obtenus avec les trois jeux de connaissances initiaux sur la zone de test (échelle logarithmique pour le nombre d'états parcourus)

Nous présentons en figure F.13 les résultats cartographiques obtenus avec nos trois jeux de connaissances sur la zone de test. Les résultats sont dans l'ensemble satisfaisants pour les trois jeux de connaissances. On peut toutefois noter des différences de qualité entre les résultats obtenus avec les trois jeux de connaissances. Les résultats obtenus avec le jeu de connaissances *basique* sont meilleurs en tout point que les autres (ce qui est logique compte tenu des spécificités de ce jeu de connaissances). Nous rappelons en effet que le jeu de connaissances *basique* permet de trouver le meilleur état (en termes de satisfaction) pour chaque groupement de bâtiments. Les résultats obtenus avec le jeu de connaissances *basique* ne comportent que de très rares problèmes de densité et de proximité. Ces rares problèmes peuvent être expliqués par la complexité de certains *groupements de bâtiments* et les limites de nos fonctions d'évaluation des contraintes. En effet, à chaque état, chaque contrainte calcule sa valeur de satisfaction. Cette valeur est un entier compris entre 1 et 10. Le nombre de valeurs possibles est donc faible et peut se montrer insuffisant pour caractériser les nombreux états différents dans lesquels peuvent se trouver les agents géographiques complexes. Concernant les deux autres jeux de connaissances, nous retrouvons bien les légers problèmes de densité et de proximité relevés pour le jeu de connaissances *basique*. Le jeu de connaissances défini par l'expert du modèle AGENT obtient des résultats assez proches de ceux obtenus avec le jeu de connaissances *basique* malgré quelques problèmes supplémentaires de densité et de proximité.
Les résultats obtenus avec le jeu de connaissances défini par l'expert en cartographie comportent plus de problèmes de densité et de proximité. Nous pouvons également constater qu'un certain nombre de bâtiments n'ont pas été généralisés.

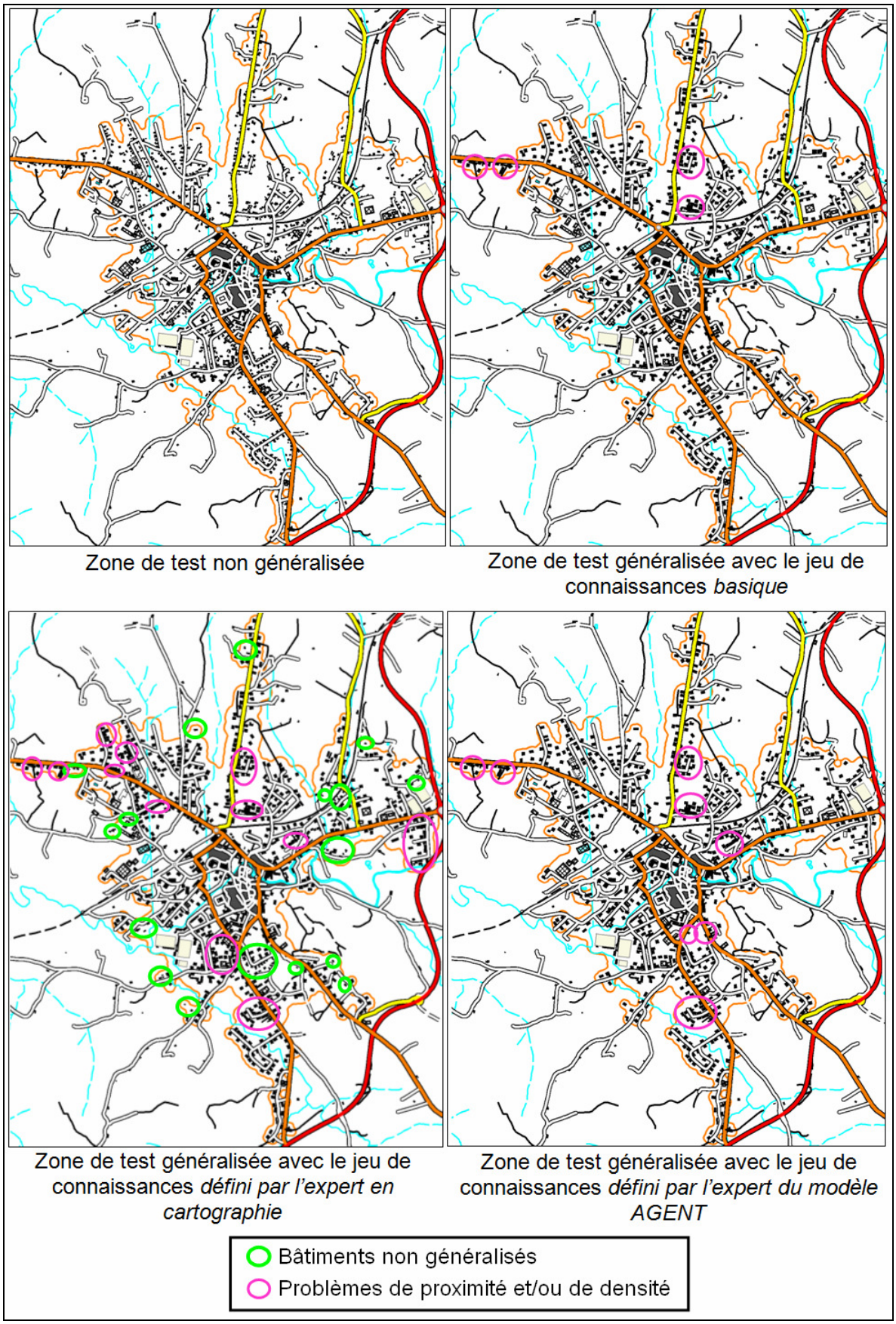


Figure F.13 Résultats cartographiques obtenus avec les trois jeux de connaissances initiaux sur la zone de test

F.II.4 Choix des algorithmes d'apprentissage et d'exploration et de leurs paramètres pour les expérimentations

Certaines des méthodes que nous avons proposées nécessitent de définir des valeurs de paramètres et de choisir des algorithmes. Nous détaillons en partie F.II.4.1 les algorithmes choisis pour nos expérimentations et en partie F.II.4.2 la fonction d'évaluation des jeux de connaissances utilisée.

F.II.4.1 Algorithmes choisis

Un certain nombre de nos approches sont basées sur l'utilisation d'algorithmes d'apprentissage ou de résolution de problèmes. La démarche générale dans le choix de ces algorithmes pour nos expérimentations a été de choisir en priorité des algorithmes classiques très couramment employés plutôt que de chercher à utiliser des algorithmes très récents.

Notre approche générale de révision des connaissances est basée sur l'utilisation d'un échantillon d'instances du problème considéré. Nous avons proposé dans ce cadre une méthode visant à sélectionner, parmi un ensemble d'instances, un échantillon représentatif (cf. C.II.2). Cette méthode est basée sur l'utilisation d'un algorithme de classification non supervisée. Nous avons utilisé pour les expérimentations un algorithme de classification non supervisée de classe EM (Dempster et al., 1977). Cette classe d'algorithmes, que nous présentons en annexe *An.2.3*, est très souvent employée pour la classification de données. Ils permettent, comme exigé par notre méthode, en plus de diviser l'ensemble des objets considérés en groupes, de définir pour chaque objet une probabilité d'appartenance à chaque groupe.

Nous avons défini au chapitre D une approche destinée à la révision des connaissances représentées sous forme de bases de règles de production. Cette approche est basée sur une première phase de partitionnement de l'espace des mesures (cf. D.III.3.4). Nous avons proposé dans ce cadre deux approches. Une première est basée sur la discrétisation indépendante des mesures (cf. D.III.3.4.2). Cette première approche nécessite de choisir un algorithme de sélection d'attributs et un algorithme de discrétisation d'attributs numériques. Nous avons utilisé comme algorithme de sélection d'attributs celui proposé par (Hall & Smith, 1998) et comme algorithme de discrétisation celui proposé par (Fayyad & Irani, 1992). Nous avons également utilisé ces deux algorithmes dans le cadre de l'évaluation des jeux de mesures (cf. C.IV.3). Nous présentons ces deux algorithmes en annexes *An.2.4* et *An.2.5*. La seconde approche de partitionnement de l'ensemble des mesures est basée sur l'utilisation d'un algorithme d'apprentissage supervisé (cf. D.III.3.4.3). Nous avons utilisé dans ce cadre l'algorithme C4.5 que nous présentons en annexe *An.2.2*. Cet algorithme, qui est l'un des algorithmes d'apprentissage symbolique les plus utilisés, a pour avantage d'être résistant au bruit.

Notre approche de révision est enfin basée sur l'utilisation d'un algorithme de recherche locale permettant d'explorer le domaine de définition des connaissances (cf. D.III.2.1). Nous avons utilisé dans ce cadre l'algorithme de recherche locale avec tabous (Glover, 1989). Nous présentons cet algorithme en annexe *An.2.6*. Cet algorithme demande de paramétrer la taille de la liste taboue. Nous avons défini cette valeur empiriquement au travers de tests menés sur d'autres données et en prenant en compte la taille maximale du voisinage. Nous avons ainsi pris une taille pour la liste taboue égale à *(taille voisinage/5)*. Nous avons choisi comme

critère d'arrêt de mettre fin à l'exploration après 2000 itérations (mouvements) sans amélioration de la meilleure solution trouvée. Ce nombre a été défini empiriquement.

F.II.4.2 Fonction d'efficacité et de performance

Notre approche de révision est basée sur l'utilisation d'une fonction d'évaluation des performances définissant, pour un système de résolution de problèmes utilisant un jeu de connaissances donné, ces performances pour un échantillon d'instances du problème considéré.

Cette fonction doit, en fonction des besoins de l'utilisateur, tenir à la fois compte de l'efficacité du système mais également de son efficience.

Dans le cadre de nos expérimentations, nous avons utilisé la fonction suivante :

$$Perf(S_K, P_n) = \frac{4 \times Efficacité(S_K, P_n) + Efficience(S_K, P_n)}{5}$$

Le facteur 4 donné à l'efficacité s'explique par le souhait de privilégier fortement l'efficacité du système par rapport à son efficience. Il ne sert à rien d'obtenir très rapidement un résultat si celui-ci est mauvais. La valeur du facteur a été définie empiriquement au travers de nombreux tests de généralisation menés sur la zone de révision.

Concernant la fonction d'évaluation de l'efficacité du système, nous proposons de reprendre la fonction *Efficacité(S_K,P_n)* que nous avons définie en partie E.IV.3 :

$$Efficacité(S_K, Pn) = \left(\frac{PremQuartile(S_K, P_n) + Moyenne(S_K, P_n)}{20} \right)^2$$

Cette fonction permet de prendre en compte la satisfaction moyenne obtenue sur l'échantillon de révision au travers de la fonction *Moyenne(S_K, P_n)* et de pondérer ce résultat par la valeur du premier quartile (*PremQuartile(S_K, P_n)*). L'intérêt de cette pondération provient du fait qu'il est préférable pour les agences cartographiques d'obtenir trois quarts de bonnes généralisations et le restant de mauvaises plutôt que d'obtenir des résultats de qualité homogène mais moyen (cf. E.IV.3). Le facteur 1/20 sert à normaliser la valeur de cette fonction. En effet, nous rappelons que la satisfaction d'un agent géographique est comprise entre 1 et 10. Nous avons ajouté un carré par rapport à la fonction présentée en partie E.IV.3 de façon à accentuer les écarts entre valeurs.

Concernant la fonction *Efficience(S_K,P_n)* d'évaluation de l'efficience du système, nous proposons la fonction suivante :

$$Efficience(S_K, P_n) = NbEtatsAvecComplexité(S_K, P_n)^{\frac{1}{3} \times \left(1 - \sqrt[5]{NbEtatsAvecComplexité(S_K, P_n)}\right)}$$

Nous donnons en figure F.14 le tracé de la fonction $f(x) = x^{\frac{1}{3}\left(1-\sqrt[5]{x}\right)}$.

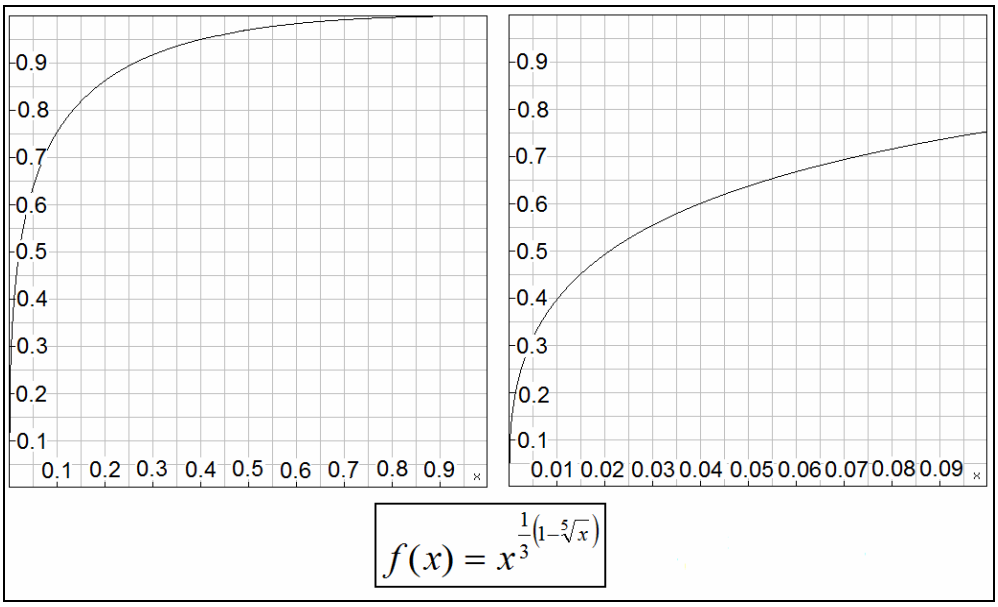


Figure F.14 Tracé de la fonction $f(x) = x^{\frac{1}{3}\left(1-\sqrt[5]{x}\right)}$ sur l'intervalle [0,1] et sur l'intervalle [0,0.1]

La fonction *NbEtatsAvecComplexité(S_K, P_n)* est une fonction qui calcule la moyenne du nombre d'états parcourus pour chaque généralisation pondérée par leur complexité a priori. En effet, le coût en termes de calcul de l'application d'une action ou de la caractérisation d'un nouvel état sera généralement dépendant du nombre de bâtiments composant le groupement de bâtiments. Ainsi, il sera beaucoup moins lourd en termes de temps de calcul de parcourir 10 états pour un groupement de bâtiments composé de 2 bâtiments que pour un groupement de bâtiments composé de 20 bâtiments. Nous proposons donc de prendre pour la fonction d'efficience, la moyenne du nombre d'états pondérée par le nombre de bâtiments concernés.

$$NbEtatsAvecComplexité(S_K, P_n) = \frac{\sum_{x \in P_n} nb\_bâtiments(x) \times nb\_état(x)}{\sum_{x \in P_n} nb\_bâtiments(x)}$$

Le choix de cette fonction d'efficience s'explique par la volonté de disposer d'une fonction permettant de couvrir une large plage de valeurs (dans l'intervalle [0,1]) pour des valeurs moyennes de nombres d'états compris entre 10 et ∞. En effet, nous avons pu définir empiriquement qu'une moyenne de 10 états parcourus par *groupement de bâtiments* était acceptable en termes d'efficience. Il n'est pas indispensable de trouver des connaissances permettant de parcourir un nombre moyen d'états plus faible (surtout si cela dégrade l'efficacité du processus).

Nous donnons dans la table suivante les valeurs prises par les différents critères pour les trois jeux de connaissances sur la zone de test :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Jeu de connaissances basique	**0.862**	0.355	0.761
Jeu de connaissances défini par l'expert en cartographie	0.807	**0.728**	0.788
Jeu de connaissances défini par l'expert du modèle AGENT	0.838	0.599	**0.79**

Ces valeurs sont cohérentes avec les résultats observés sur les nuages de points et sur les résultats cartographiques. Le jeu de connaissances *basique* permet d'obtenir la plus forte valeur d'efficacité mais obtient un score faible d'efficience, ce qui lui donne au final une valeur de performance faible comparée aux deux autres jeux de connaissances. Ces deux derniers obtiennent de biens meilleurs scores d'efficience. A cet égard, c'est le jeu de connaissances *défini par l'expert en cartographie* qui obtient le meilleur score d'efficience. Le score d'efficacité obtenu avec celui-ci est par contre inférieur à celui obtenu avec le jeu de connaissances *défini par l'expert du modèle AGENT*, ce qui explique au final son score de performance plus faible.

Nous pouvons néanmoins reprocher à notre fonction d'évaluation des performances de fournir des valeurs trop proches pour des jeux de connaissances dont les qualités « réelles » sont plus nuancées.

Un premier problème vient de la fonction d'évaluation de l'efficacité. Les valeurs obtenues par les trois jeux de connaissances sont relativement proches alors que les résultats cartographiques sont beaucoup plus nuancés. Ainsi, la valeur d'efficacité obtenue par le jeu de connaissances *défini par l'expert en cartographie* n'a qu'une différence de 0.05 avec la valeur obtenue par le jeu de connaissances *basique* alors que les différences cartographiques entre les deux jeux de connaissances sont importantes. Nous pouvons expliquer ce problème par la plage de valeurs utilisée pour caractériser la satisfaction d'un agent géographique. En théorie cette valeur devrait en effet varier entre 1 et 10. En pratique elle varie entre 6 et 10. Effectivement, le calcul de la valeur de satisfaction d'un agent géographique prend en compte la satisfaction de l'ensemble des contraintes (moyenne des satisfactions des contraintes pondérée par les importances). Or la plupart des agents géographiques à l'image des agents *groupement de bâtiments* disposent de nombreuses contraintes de conservation (répartition spatiale, conservation des bâtiments importants, etc.) qui ont toujours une valeur de 10 à l'état initial et qui ont souvent un poids important dans le calcul de la satisfaction. Ainsi, pour donner un exemple, la valeur de satisfaction d'un agent *groupement de bâtiments* à l'état initial est généralement proche de 7.

Un second problème provient de la fonction utilisée pour agréger les valeurs d'efficience et d'efficacité. Nous avons utilisé une moyenne pondérée pour cette agrégation pour notre fonction d'évaluation des performances. Ce type de fonction d'agrégation est assez limité et ne permet pas de prendre en considération des cas limites. Ainsi, le jeu de connaissances *basique* devrait avoir un score de performance faible car il nécessite de parcourir un nombre d'états beaucoup trop important pour pouvoir être utilisé dans le cadre d'une application réelle. Le jeu de connaissances *défini par l'expert en cartographie* devrait lui aussi avoir un score faible car les résultats cartographiques obtenus avec celui-ci sont insuffisants. Le jeu de

connaissances *défini par l'expert du modèle AGENT* devrait avoir un score de performance largement supérieur à celui des deux autres.

En conclusion, la fonction d'évaluation des performances, que nous avons proposée, reste cohérente mais pourrait être largement améliorée. Nous reviendrons sur les problèmes soulevés par la définition d'une telle fonction en partie F.VII.2.

F.II.5 Implémentation

Les méthodes proposées dans ce travail de thèse ont été implémentées dans le SIG Clarity™. Ce SIG est utilisé comme plate-forme pour les recherches en généralisation au laboratoire COGIT de l'IGN. Il a été implémenté à l'aide de trois langages de programmation (C, Lull et Java). Il permet d'intégrer facilement de nouveaux éléments implémentés en Lull (un langage propriétaire) et en Java.

Pour des raisons de pérennité du code et de richesse des bibliothèques de développement disponibles, notre choix s'est porté sur le langage Java pour l'implémentation de nos méthodes.

Nous avons également réimplémenté le modèle AGENT afin que celui-ci intègre les enrichissements proposés en partie B.V.3.1 et qu'il soit compatible avec notre modèle général d'organisation des connaissances (cf. B.V.2).

Concernant l'implémentation des algorithmes d'apprentissage, de sélection d'attributs et de discrétisation d'attributs, nous avons utilisé la bibliothèque Open Source Weka (Witten & Frank, 2005).

F.III Expérimentation du modèle de révision des connaissances procédurales

F.III.1 Introduction

Cette première série d'expérimentations est destinée à tester notre approche générale de révision des connaissances ainsi que l'ensemble des approches proposées dans le cadre de la phase d'analyse (cf. D.IV).

Nous proposons à cet effet de réviser dans un premier temps chaque type de connaissances du modèle AGENT indépendamment les unes des autres. Ces tests nous permettront d'évaluer notre approche générale de révision pour chacun des types de connaissances. Nous testerons dans un second temps notre approche générale de révision de l'ensemble des connaissances.

Dans le cadre de la révision d'un type de connaissances, nous avons proposé deux approches de *révision par analyse*. La première est une approche générique qui consiste à explorer directement les domaines de définition des connaissances afin de déterminer quelles valeurs de ces domaines maximisent la fonction d'évaluation (cf. D.III.2.1). Nous avons proposé d'appliquer cette approche pour les connaissances relatives à la validité des états (cf. D.III.2.4.1) et les connaissances relatives à la restriction d'application des actions (cf. D.III.2.4.2).La seconde approche est destinée à un type spécifique de bases de règles de production (cf. D.III.3). Elle est composée de trois étapes : la première consiste à partitionner l'espace des jeux de mesures en zones admettant une conclusion a priori homogène, la seconde à chercher la meilleure affectation de conclusion possible pour chacune de ces zones et la dernière à agréger des zones afin de simplifier les bases de règles obtenues. Nous avons proposé d'appliquer cette seconde approche pour les connaissances relatives à l'optimalité des états, à la fin de cycle, à l'application des actions et la priorité des contraintes.

Nous proposons également d'étudier l'influence du *coefficient de qualité des connaissances* sur leur révision. Nous rappelons qu'un coefficient est défini par connaissance et qu'il caractérise la qualité de la valeur définie pour la connaissance. Plus il est élevé, moins la valeur de sa connaissance sera susceptible d'être modifiée lors du déclenchement d'un processus de révision des connaissances. Nous proposons de tester pour chaque type de connaissances trois valeurs pour ce coefficient : 0 (prise en compte minimale des connaissances initiales), 0.01 (prise en compte moyenne des connaissances initiales) et 0.1 (prise en compte forte des connaissances initiales). Ces valeurs ont été définies empiriquement par des tests menés sur la zone de révision avec d'autres jeux de connaissances que ceux utilisés pour ces expérimentations.

Nous présentons en partie F.III.2 les expérimentations consacrées à la révision de la connaissance relative à la validité des états. Les résultats de ces expérimentations permettront de donner une première évaluation de notre approche générale de révision des connaissances. L'approche générale comprend ici la méthode de sélection automatique de l'échantillon de révision et une exploration exhaustive du domaine de définition de la connaissance. Nous pourrons également fournir une première analyse de l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

Nous présentons en partie F.III.3 les expérimentations consacrées à la révision des connaissances relatives à la restriction sur les actions. Les résultats de ces expérimentations permettent de fournir une seconde évaluation de notre approche générale de révision des connaissances. Comme pour la partie F.II, l'approche générale est ici aussi composée de la méthode de sélection automatique de l'échantillon de révision et d'une exploration exhaustive du domaine de définition des connaissances. Nous analysons également l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

Nous présentons en partie F.III.4 les expérimentations consacrées à la révision de la connaissance relative à l'optimalité des états. Les résultats de ces expérimentations permettent de fournir une nouvelle évaluation de notre approche générale de révision des connaissances. Cette dernière s'appuie sur notre approche de *révision par analyse* spécialisée pour les connaissances représentées sous forme de bases de règles de $\mathcal{BR}$ (cf. D.III.3). Nous proposerons dans ce cadre d'évaluer les résultats obtenus avec nos deux approches de partitionnement. Nous proposons également de tester l'influence de la phase de simplification des règles dans les résultats obtenus. Nous analysons enfin l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

Nous présentons en partie F.III.5 les expérimentations consacrées à la révision de la connaissance relative à la fin de cycle. Comme pour la connaissance relative à l'optimalité des états, nous testerons, en plus de notre approche générale de révision, nos deux approches de partitionnement ainsi que l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

Nous présentons en partie F.III.6 les expérimentations consacrées à la révision des connaissances relatives à la priorité des contraintes. Les résultats de ces expérimentations nous permettront d'évaluer, en plus de notre approche générale de révision, nos différentes approches d'exploration de l'espace des affectations de conclusion possibles. Nous analyserons aussi l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

Nous présentons en partie F.III.7 les expérimentations consacrées à la révision des connaissances relatives à l'application des actions. Comme pour les connaissances relatives à la priorité des contraintes, nous testerons, en plus de notre approche générale de révision, nos différentes approches d'exploration de l'espace des affectations de conclusion possibles. Nous analyserons également l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

Enfin, la partie F.III.8 présente les expérimentations consacrées la révision complète des connaissances. Ces expérimentations permettront d'évaluer notre approche de révision séquentielle des connaissances. Nous proposerons également une analyse poussée de l'influence du *coefficient de qualité des connaissances* sur le processus de révision.

F.III.2 Révision de la connaissance relative à la validité des états

F.III.2.1 Contexte de l'expérimentation

Nous avons donné la possibilité, parmi les enrichissements proposés pour le modèle AGENT (cf. B.V.3.1), de choisir le critère de validité parmi six critères prédéfinis.

L'objectif de la révision consiste à définir lequel de ces six critères, est le plus pertinent pour le type d'agent géographique considéré.

En raison de la taille réduite du domaine de définition de cette connaissance, nous avons proposé en partie D.IV.2.4.1 de réviser cette connaissance en testant chacun des six critères prédéfinis. Le critère conservé est celui maximisant la fonction d'évaluation définie en partie D.IV.2.2. La fonction de performance utilisée par la fonction d'évaluation est celle proposée en partie F.II.3.3. Nous rappelons que cette fonction d'évaluation dépend également du taux de modification de la connaissance que nous avons défini en partie D.IV.2.4.1.

Cette première expérimentation permet de donner une première évaluation de notre approche générale de révision. Elle permet également de donner des informations sur les critères de validité prédéfinis.

Nous rappelons que les critères de validité définis dans les jeux de connaissances initiaux sont :

- Jeu de connaissances basique : ***CEPAC***
- Jeu de connaissances défini par l'expert en cartographie : ***CEPAC ET CEEAP***
- Jeu de connaissances défini par l'expert du modèle AGENT : ***CEPAC ET CEEAP***

Le jeu de connaissances *basique* admet pour critère de validité le critère ***CEPAC***. Nous rappelons que ce critère est celui utilisé par les jeux de connaissances *minimaux*. Il s'agit du critère qui induit le moins d'élagage. Il ne sera donc pas possible d'améliorer l'efficacité du système de généralisation par une révision de la connaissance de validité du jeu de connaissances *basique*. Il sera par contre possible d'améliorer son efficience et donc ses performances générales.

F.III.2.2 Résultats obtenus

Le critère de validité obtenu après révision a été, pour les trois jeux de connaissances, le critère ***CEPAC ET CEEAP*** (le critère de base de Clarity™), pour les trois valeurs testées du *coefficient de qualité de la connaissance*.

A titre comparatif nous présentons, dans les tables suivantes, les performances obtenues par chaque critère d'élagage sur l'échantillon de révision.

Résultats des différents critères pour le jeu de connaissances *basique* :

Critère d'élagage	***Efficacité(S_K,P_n)***	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***
CEPAC	**0.841**	0.522	0.748
CEPAC ET CEENS	0.827	0.533	0.768
CEPAC ET CEEAP	0.809	0.623	**0.772**
CEPAC ET CEPAG	0.785	0.522	0.732
CEPAC ET CEPAG ET CEENS	0.763	0.624	0.735
CEPAC ET CEPAG ET CEEAP	0.755	**0.675**	0.739

Résultats des différents critères pour le jeu de connaissances *défini par l'expert en cartographie* :

Critère d'élagage	***Efficacité(S_K,P_n)***	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***
CEPAC	**0.802**	0.497	0.741
CEPAC ET CEENS	0.797	0.597	0.757
CEPAC ET CEEAP	0.795	0.681	**0.772**
CEPAC ET CEPAG	0.77	0.635	0.743
CEPAC ET CEPAG ET CEENS	0.745	0.687	0.733
CEPAC ET CEPAG ET CEEAP	0.739	**0.736**	0.739

Résultats des différents critères pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

Critère d'élagage	***Efficacité(S_K,P_n)***	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***
CEPAC	**0.84**	0.408	0.753
CEPAC ET CEENS	0.82	0.545	0.765
CEPAC ET CEEAP	0.811	0.62	**0.773**
CEPAC ET CEPAG	0.784	0.545	0.736
CEPAC ET CEPAG ET CEENS	0.754	0.632	0.73
CEPAC ET CEPAG ET CEEAP	0.749	**0.677**	0.735

Nous constatons que le critère ***CEPAC ET CEEAP*** permet d'obtenir de meilleurs résultats que les cinq autres critères pour les trois jeux de connaissances. En effet, ce critère permet un très bon compromis entre efficience et efficacité.

Concernant le jeu de connaissance basique, nous avons testé ses performances avec son critère de validité initial et son critère révisé sur la zone de test. Les résultats obtenus sont présentés en figure F.15 et dans la table suivante :

Critère d'élagage	***Efficacité(S_K,P_n)***	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***
Avant révision	0.862	0.355	0.761
Après révision	0.837	0.599	0.79

Ces résultats montrent que le changement de critère de validité a permis une diminution très importante du nombre d'états parcourus pour la très grande majorité des agents *groupement de bâtiments*. Ainsi, le score d'efficience a fortement augmenté, ce qui a permis, malgré une légèrement diminution de l'efficacité du système, d'améliorer de manière conséquente les performances du système selon notre critère.

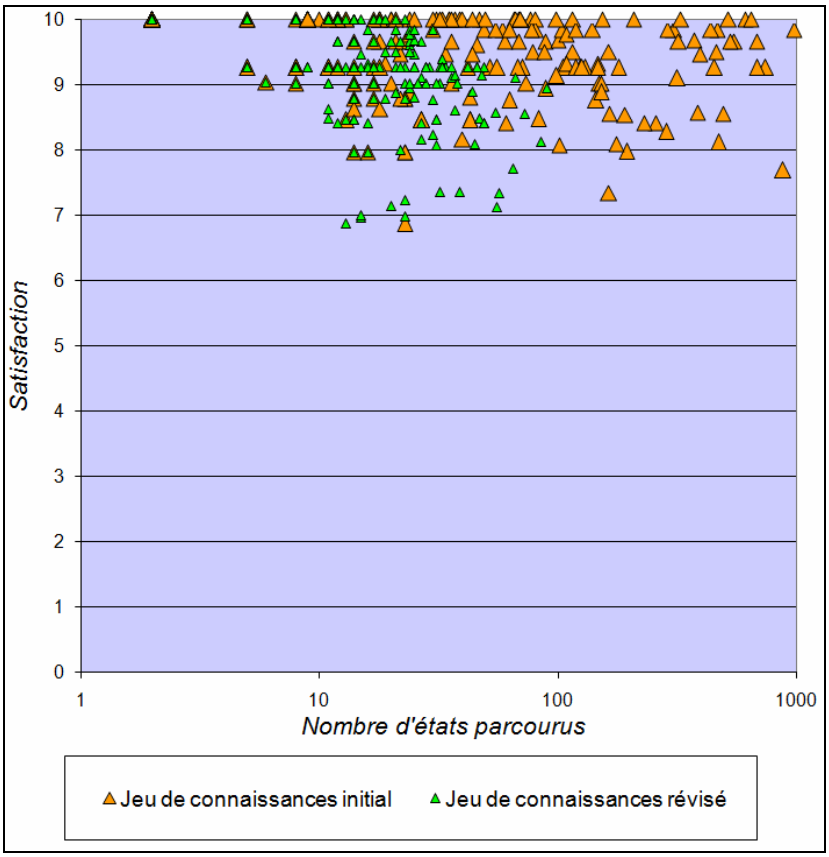


Figure F.15 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision de la connaissance relative à la validité des états (échelle logarithmique pour le nombre d'états parcourus)

Les résultats cartographiques obtenus sur la zone de test après révision (figure F.16) sont globalement proches de ceux obtenus avant révision. Quelques groupements de bâtiments ont été légèrement moins bien généralisés (avec des problèmes de densité et de proximité).

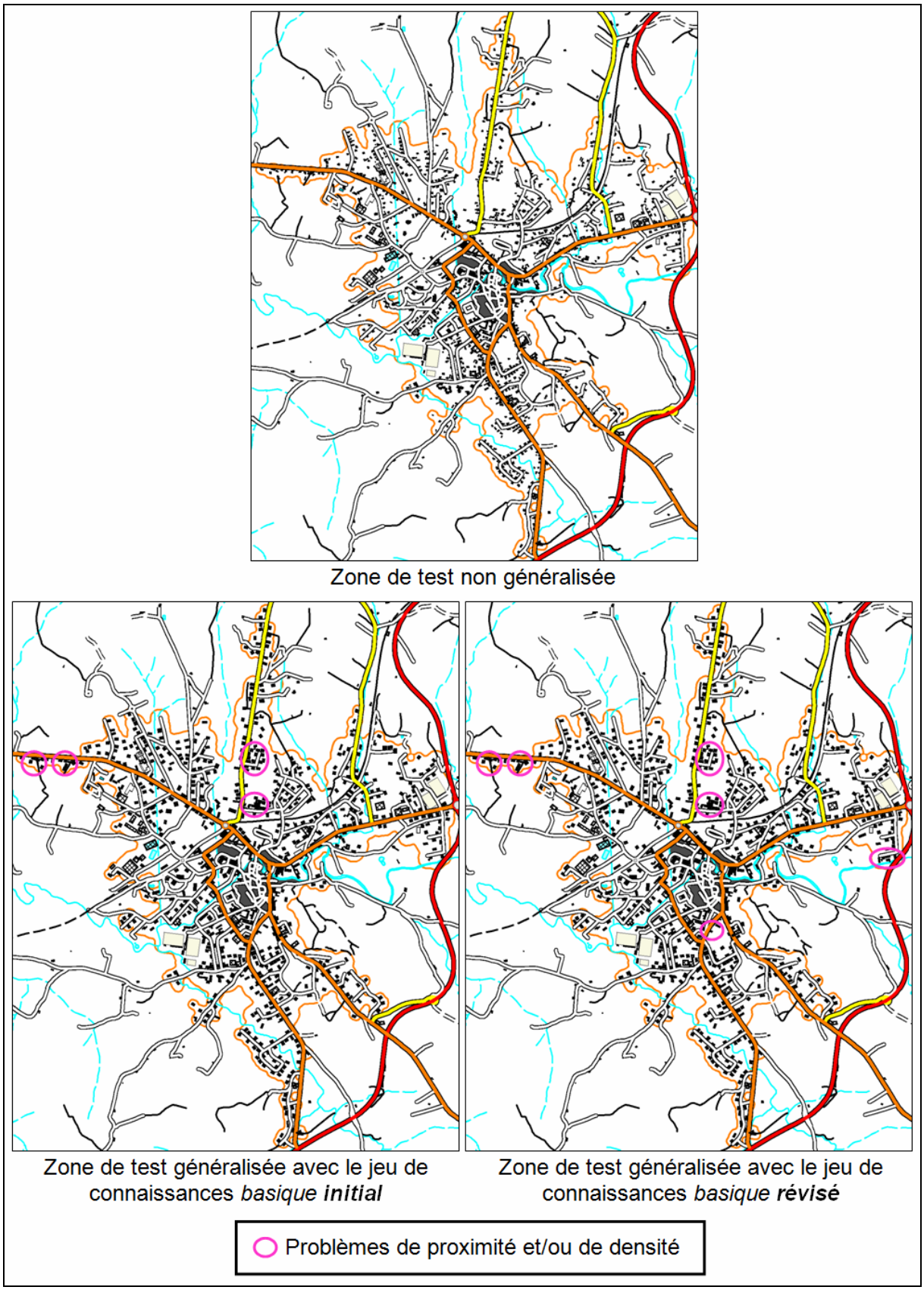


Figure F.16 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision de la connaissance relative à la validité des états

F.III.2.3 Bilan de l'expérimentation

Cette première expérimentation a permis de montrer que notre approche de révision de la connaissance relative au choix de critère de validité permet d'améliorer un jeu de connaissances.

Elle a également mis en lumière que le critère ***CEPAC ET CEEAP*** (critère de base implémenté dans Clarity™) propose un compromis particulièrement intéressant entre efficience et efficacité. Parmi les autres critères, seul le critère ***CEPAC ET CEENS*** permet d'obtenir des scores de performance proche de ceux obtenus avec le critère ***CEPAC ET CEEAP***.

Le *coefficient de qualité de la connaissance* n'a eu aucune influence lors de cette expérimentation. En effet, le seul jeu de connaissances pour lequel le processus de révision a modifié la valeur du critère de validité est le jeu de connaissances basique qui est passé du critère ***CEPAC*** au critère ***CEPAC ET CEEAP***. Or, les expérimentations ont montré que pour ce jeu les résultats étaient largement meilleurs avec le critère ***CEPAC ET CEEAP*** qu'avec le critère ***CEPAC***. Cela a justifié le choix du critère ***CEPAC ET CEEAP*** pour le processus de révision même pour un *coefficient de qualité de la connaissance* élevé.

F.III.3 Révision des connaissances relatives à la restriction d'application des actions

F.III.3.1 Contexte de l'expérimentation

Nous rappelons qu'il est nécessaire de définir pour chaque connaissance un nombre limite d'applications compris entre 0 et NB_MAX. Nous avons proposé en partie D.IV.2.4.2 de réviser cette connaissance par l'intermédiaire d'un test exhaustif de l'ensemble des valeurs possibles pour ces restrictions et de garder celle maximisant la fonction d'évaluation sur l'échantillon de révision.

Nous rappelons que cette connaissance a un rôle d'élagage. Le jeu de connaissances *basique* ne comporte pas de restrictions sur les actions (la valeur de restriction est égale à 10 pour toutes les actions). La révision de ces connaissances de restriction ne pourra donc qu'entraîner une baisse d'efficacité pour ce jeu de connaissances. Elle pourra par contre permettre une hausse de son efficience.

Pour les deux autres jeux de connaissances, des restrictions sont définies. Néanmoins, ces dernières étant très peu sévères, la révision a peu de chance de permettre un gain d'efficacité du système de généralisation.

L'objectif de la révision de cette connaissance sera donc d'améliorer le plus possible l'efficience du système de généralisation en essayant de limiter au maximum les pertes d'efficacité.

F.III.3.2 Résultats obtenus

Le tableau suivant présente les restrictions obtenues après révision. Le « CQ » désigne la valeur du *coefficient de qualité* définie pour les connaissances relatives à la restriction d'application des actions pour la révision.

		Action de suppression globale de bâtiments	Action de déplacement de bâtiments	Action de suppression locale de bâtiments	Action de suppression/ recentrage d'un bâtiment	Action de généralisation des sous agents
Jeu de connaissances basique	Avant révision	10	10	10	10	10
	Révision avec CQ = 0	0	3	1	0	1
	Révision avec CQ = 0.01	2	6	2	2	2
	Révision avec CQ = 0.1	10	10	10	10	10
Jeu de connaissances défini par l'expert en cartographie	Avant révision	2	10	10	10	10
	Révision avec CQ = 0	1	5	2	0	2
	Révision avec CQ = 0.01	1	10	10	10	10
	Révision avec CQ = 0.1	2	10	10	10	10
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	1	10	10	10	1
	Révision avec CQ = 0	0	2	1	0	1
	Révision avec CQ = 0.01	0	10	10	10	1
	Révision avec CQ = 0.1	1	10	10	10	1

Nous pouvons déjà constater que le coefficient de qualité des connaissances joue bien son rôle de limiteur de modifications des connaissances initiales : plus sa valeur est élevée, plus les valeurs des restrictions sont proches de celles initiales.

Le processus de révision semble dans l'ensemble avoir privilégié l'application des actions de déplacement de suppression locale de bâtiments pour la résolution des problèmes de proximité. Il semble également indiquer que l'action de suppression globale n'est pas forcément indispensable au système de généralisation. Une explication pour cette restriction est que d'autres actions proposent également des suppressions de bâtiments (action de

suppression locale de bâtiments, action de suppression/recentrage) qui peuvent permettre d'améliorer la satisfaction de la contrainte de densité. L'utilisation de l'action de suppression globale ne semble donc pas toujours indispensable.

Concernant l'action de généralisation des bâtiments, si sa limitation à une application semble logique, nous pouvons expliquer le fait qu'il soit parfois nécessaire de l'appliquer deux fois pour parfaitement généraliser des bâtiments en remarquant que le critère de validité utilisé pour généraliser les bâtiments individuellement est ***CEPAC ET CEEAP*** (critère de base de Clarity™). La figure F.17 illustre ce problème : sur cet exemple, une première généralisation a permis à l'agent *bâtiment* de trouver comme meilleur état, l'état 7. Nous posons que l'état 8 est invalide à cause de l'état 5. Si l'on active à nouveau l'agent *bâtiment*, son état initial sera cette fois l'ancien état 7 et cette fois l'état 8 sera valide car l'agent ne prendra plus en considération l'état 5. L'agent pourra donc potentiellement trouver un meilleur état que l'état 7 (ici l'état 9).
Il est donc possible d'améliorer la satisfaction d'un agent *bâtiment* en l'activant plusieurs fois et il peut donc être utile de lancer plusieurs fois l'action de généralisation des bâtiments.

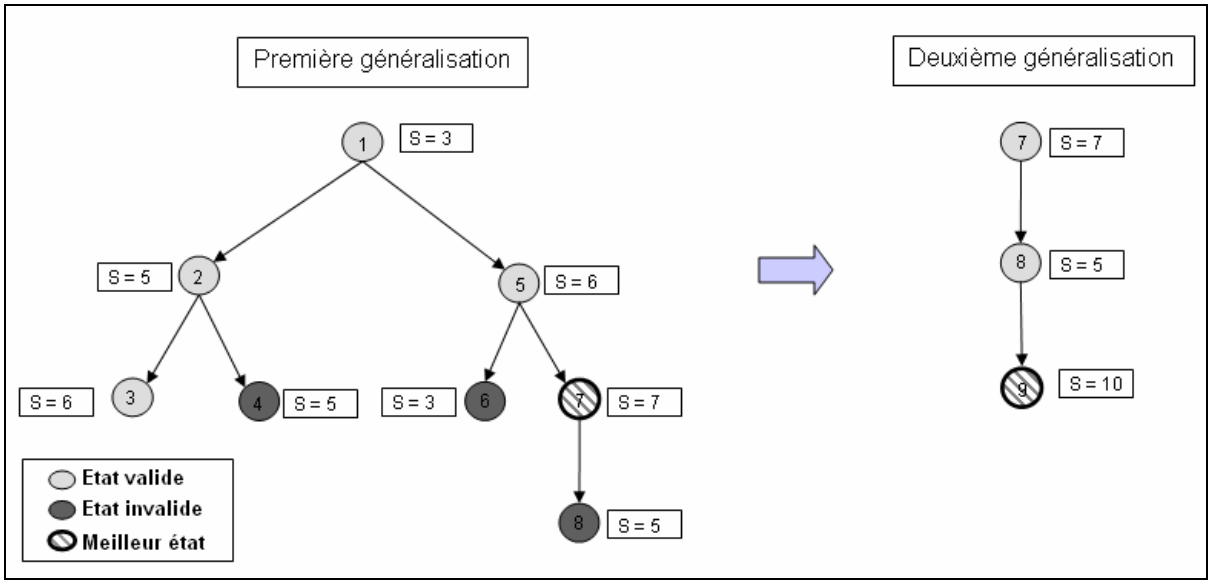


Figure F.17 Amélioration de la généralisation d'un bâtiment par la réactivation de l'agent *bâtiment*

Les résultats (figure F.18, F.19 et F.20) obtenus avec les jeux de connaissances révisés avec un *coefficient de qualité* nul montrent une forte amélioration de l'efficience du système de généralisation pour les trois jeux de connaissances initiaux. Ainsi, nous pouvons constater pour nos trois jeux de connaissances que la révision des restrictions d'application des actions a permis de limiter efficacement les cas où beaucoup d'états étaient parcourus.

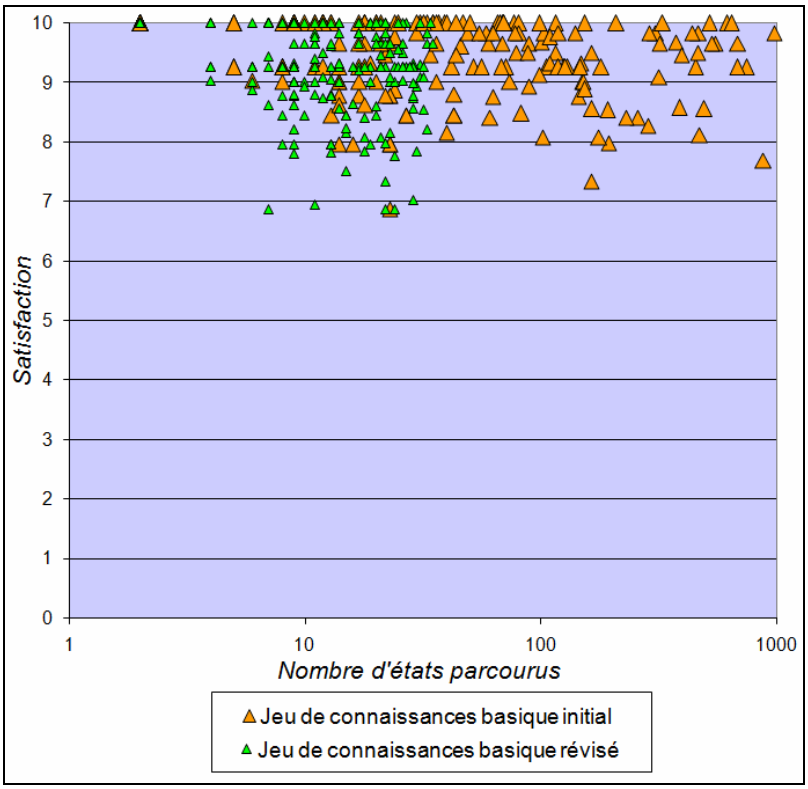


Figure F.18 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) des connaissances relatives à la restriction d'application des actions (échelle logarithmique pour le nombre d'états parcourus)

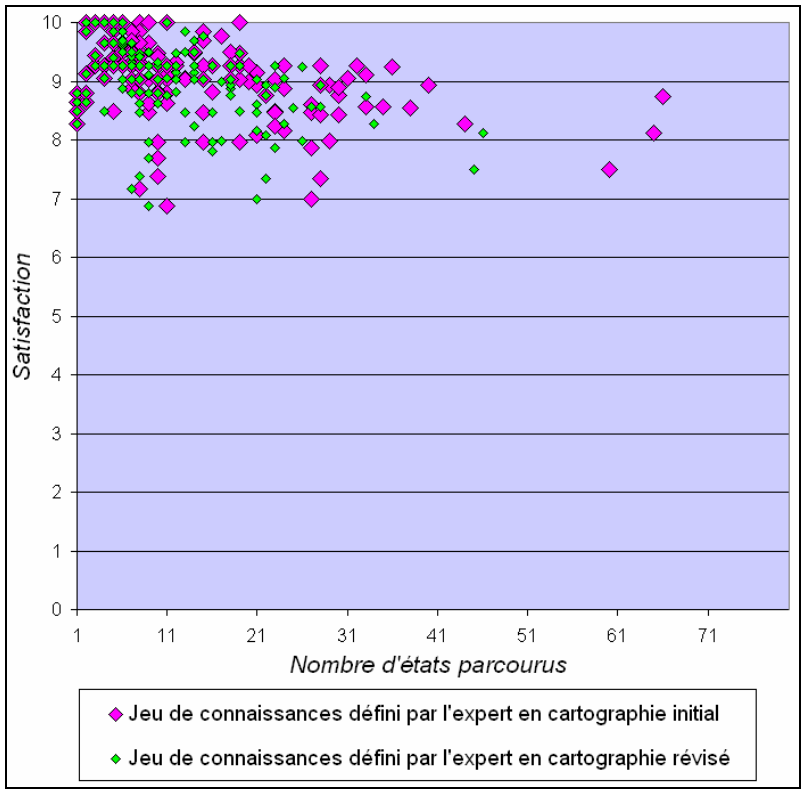


Figure F.19 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) des connaissances relatives à la restriction d'application des actions

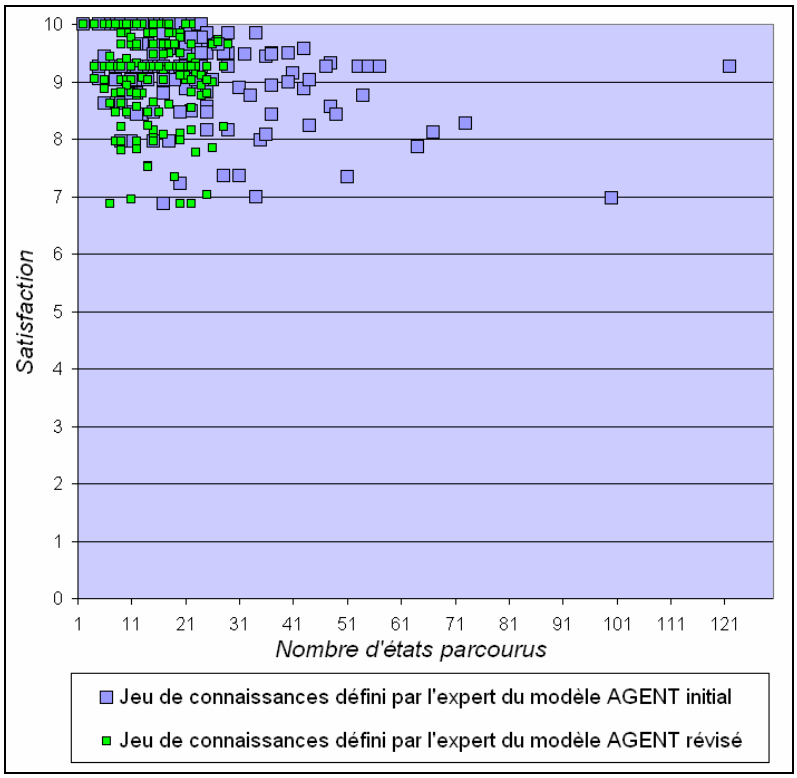


Figure F.20 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (avec CQ = 0) des connaissances relatives à la restriction d'application des actions

Concernant les résultats cartographiques obtenus sur la zone de test avant et après révision par les connaissances (avec un coefficient de qualité des connaissances de 0), nous pouvons observer figure F.21, F.22 et F.23 que les jeux de connaissances révisés permettent d'obtenir des résultats proches des résultats initiaux. Ainsi, il n'y a pas de différences apparentes sur la figure F.22 entre les résultats obtenus avant et après révision par le jeu de connaissances *défini par l'expert en cartographie*. Concernant les deux autres jeux de connaissances (figure F.21 et F.23), si l'on peut noter l'apparition de quelques problèmes de proximité ou de densité supplémentaires, les résultats restent très satisfaisants.

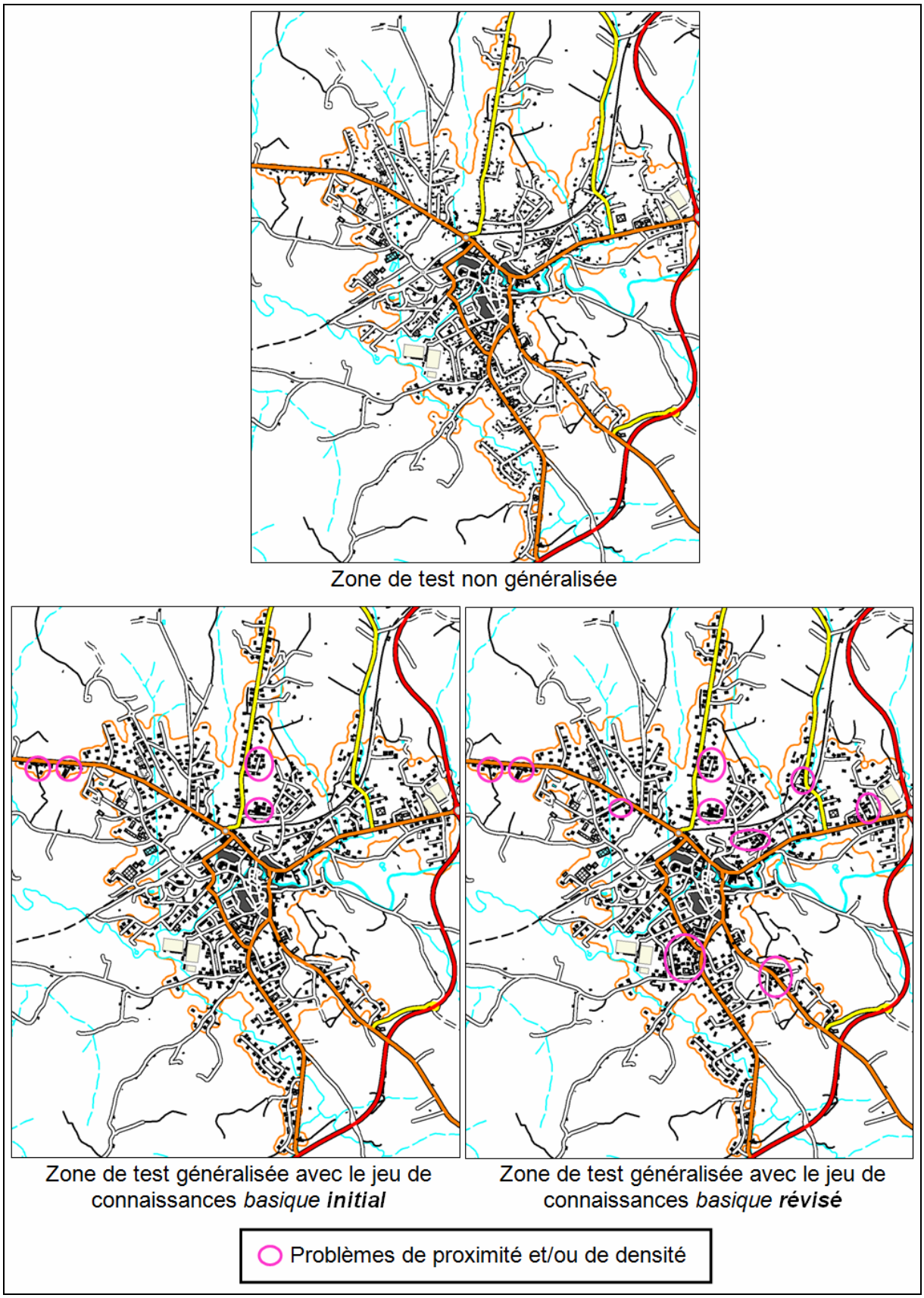


Figure F.21 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) des connaissances relatives à la restriction d'application des actions

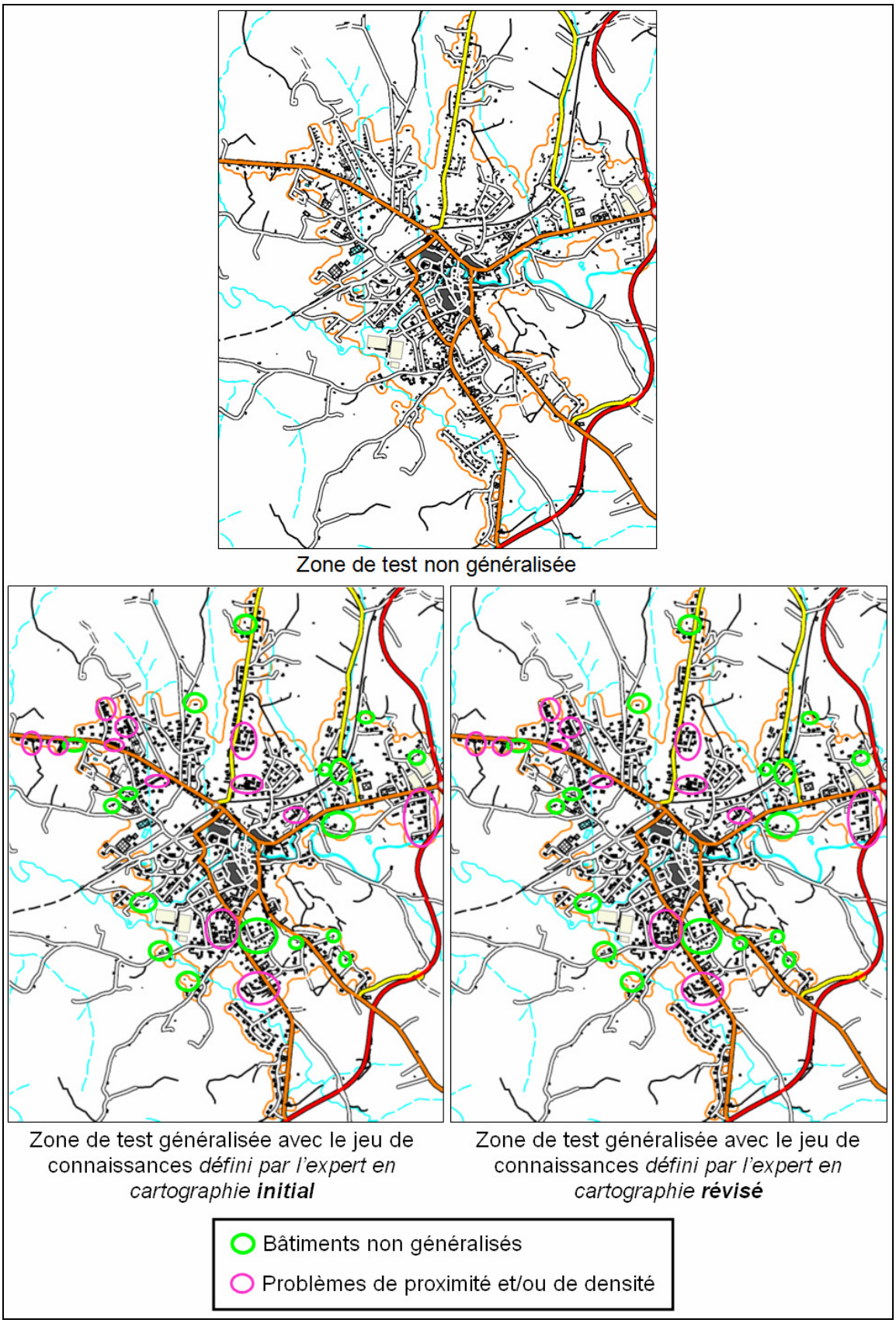


Figure F.22 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) des connaissances relatives à la restriction d'application des actions

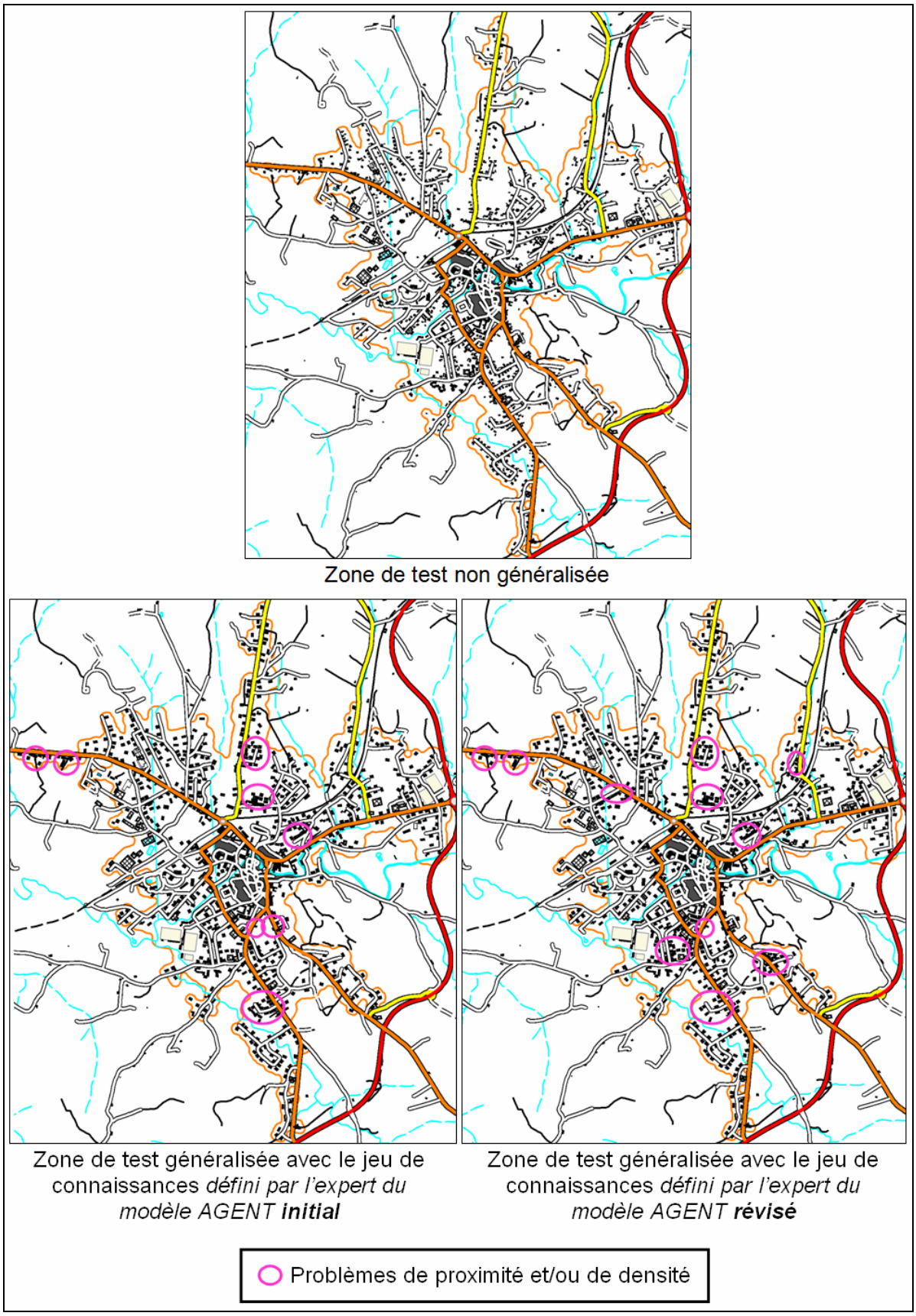


Figure F.23 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (avec CQ = 0) des connaissances relatives à la restriction d'application des actions

Les tables suivantes donnent les résultats des jeux de connaissances obtenus sur la zone de test.

Révision des connaissances relatives à la restriction d'application d'une action pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.862**	0.355	0.761
Après révision avec ***CQ*** *= 0*	0.824	**0.671**	**0.794**
Après révision avec ***CQ*** *= 0.01*	0.847	0.458	0.769
Après révision avec ***CQ*** *= 0.1*	**0.862**	0.355	0.761

Révision des connaissances relatives à la restriction d'application d'une action pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.807**	0.69	0.784
Après révision avec ***CQ*** *= 0*	0.803	**0.728**	**0.788**
Après révision avec ***CQ*** *= 0.01*	**0.807**	0.693	0.785
Après révision avec ***CQ*** *= 0.1*	**0.807**	0.69	0.784

Révision des connaissances relatives à la restriction d'application d'une action pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.838**	**0.599**	0.79
Après révision avec ***CQ*** *= 0*	0.819	**0.690**	**0.794**
Après révision avec ***CQ*** *= 0.01*	0.828	0.636	0.79
Après révision avec ***CQ*** *= 0.1*	**0.838**	0.599	0.79

Ces résultats confirment les observations que nous avons pu faire sur les nuages de points et sur les résultats cartographiques. La révision des restrictions d'application des actions a permis d'augmenter le score d'efficience obtenu par les trois jeux de connaissances et cela au détriment d'une baisse plus ou moins importante de l'efficacité. Ainsi, c'est pour le jeu de connaissances *basique* dont le score d'efficacité est le plus élevé que l'efficacité a le plus baissé mais c'est aussi pour ce jeu de connaissances que le score d'efficience a le plus augmenté, permettant une hausse importante du score de performance. Concernant le jeu de connaissances *défini par l'expert en cartographie*, le score d'efficacité n'a presque pas diminué (expliquant le fait que les différences cartographiques entre les deux résultats sont minimes) et celui d'efficience a légèrement progressé d'où un gain du score de performance. Le jeu de connaissances *défini par l'expert du modèle AGENT* a vu son score d'efficacité diminuer, mais en raison d'une augmentation importante de son score d'efficience. La révision des restrictions a également permis pour ce jeu d'améliorer ses performances.

Les résultats obtenus avec les différentes valeurs du *coefficient de qualité* des connaissances initiales sont cohérents avec les restrictions obtenues après révision pour ces différentes valeurs. Ainsi, plus le *coefficient de qualité* est élevé, plus les restrictions sont proches de celles initiales et donc plus le jeu de connaissances a des scores d'efficacité et d'efficience proches de ceux obtenus initialement.

F.III.3.3 Bilan de l'expérimentation

Cette expérimentation a permis de valider l'application de notre approche de révision pour la révision des restrictions d'application des actions. Nous avons en effet pu observer une amélioration des performances de nos trois jeux de connaissances en révisant ces restrictions d'application.

Pour nos trois jeux de connaissances, la révision des connaissances relatives à la restriction d'application des actions a permis de limiter efficacement le nombre d'états parcourus tout en ayant un impact mesuré sur l'efficacité.

Cette expérimentation a également permis de mettre en évidence le rôle du *coefficient de qualité* : plus il est élevé, moins la valeur de sa connaissance associée est modifiée par le processus de révision.

F.III.4 Révision de la connaissance relative à l'optimalité des états

F.III.4.1 Contexte de l'expérimentation

Nous rappelons qu'un état optimal représente un état qu'il n'est a priori pas possible d'améliorer compte tenu des actions disponibles (cf. B.V.3.1.1). La connaissance relative à l'optimalité des états est représentée sous la forme d'une base de règles de $\mathcal{BR}$ (cf. D.IV.3.2).

Nous appliquons pour cette connaissance l'approche proposée en partie D.IV qui est spécialement dédiée aux connaissances représentées dans un tel formalisme. Nous rappelons que cette approche est composée de trois phases (figure F.24) : dans une première phase, le processus de révision partitionne l'espace des mesures en zones admettant une conclusion a priori homogène, puis, dans une seconde phase, le processus recherche la meilleure affectation possible pour chaque zone et enfin, dans une dernière phase, le système simplifie les bases de règles obtenues par agrégation de règles (cf. D.IV.3.3).

Nous testons pour la révision des règles d'optimalité, les deux approches de partitionnement que nous avons proposées. La première approche est basée sur la discrétisation indépendante de chacune des mesures (cf. D.IV.3.4.2). La seconde approche est basée sur l'apprentissage artificiel de nouvelles règles et sur la comparaison de ces règles avec les règles initiales (cf. D.IV.3.4.3).

Nous utilisons, comme indiqué en partie F.II.3.1, l'algorithme de recherche locale taboue pour la recherche des meilleures affectations possibles de conclusion.

Enfin, nous testons les résultats obtenus avec et sans la phase de simplification des règles pour étudier l'apport de cette phase.

Nous rappelons que cette connaissance a un rôle d'élagage. Sa révision ne pourra donc pas permettre de gain d'efficacité pour le jeu de connaissances *basique* pour lequel aucune règle d'optimalité n'est définie et qui utilise le critère d'élagage ***CEPAC***. Elle a également peu de chances d'améliorer l'efficacité obtenue avec le jeu de connaissances *défini par l'expert du modèle AGENT* qui ne comporte aucune règle d'optimalité. Le seul jeu de connaissances pour lequel la révision de cette connaissance a une chance d'améliorer l'efficacité est le jeu de connaissances *défini par l'expert en cartographie* qui comporte une règle d'optimalité.

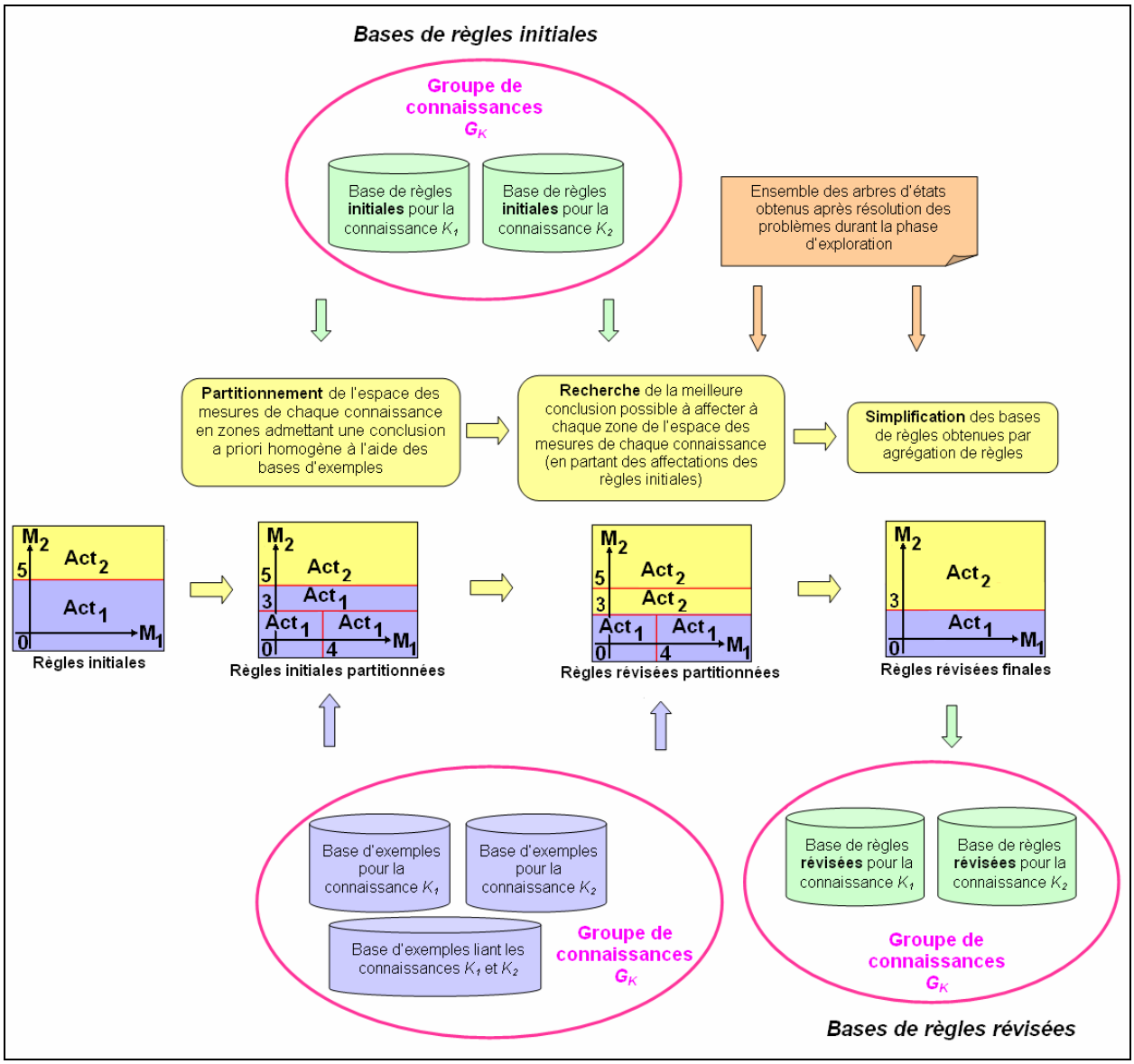


Figure F.24 Approche de révision par analyse des connaissances représentées sous forme de bases de règles de $\mathcal{BR}$

F.III.4.2 Résultats obtenus

Le tableau suivant présente les règles d'optimalité obtenues après révision avec l'approche de partitionnement par apprentissage et comparaison de règles.

		Règles d'optimalité des états
Jeu de connaissances basique	Avant révision	état toujours *non optimal*
	Révision avec CQ = 0	**Si** ($Satisfaction_{BatCoins} \leq 4$) **alors** *état optimal* **Sinon** *état non optimal*
	Révision avec CQ = 0.01	**Si** ($Satisfaction_{BatCoins} \leq 4$) **alors** *état optimal* **Sinon** *état non optimal*
	Révision avec CQ = 0.1	état toujours *non optimal*
Jeu de connaissances défini par l'expert en cartographie	Avant révision	- **Si** ($Satisfaction_{proximité} > 7$) **et** ($Satisfaction_{densité} > 7$) **et** ($Satisfaction_{satisfactionBat} > 5$) **alors** *état optimal* - **Sinon** *état non optimal*
	Révision avec CQ = 0	- **Si** ($Satisfaction_{BatCoins} \leq 6$) **alors** *état optimal* - **Sinon** *état non optimal*
	Révision avec CQ = 0.01	- **Si** ($Satisfaction_{proximité} > 7$) **et** ($Satisfaction_{densité} > 7$) **et** ($Satisfaction_{satisfactionBat} > 5$) **alors** *état optimal* - **Sinon** *état non optimal*
	Révision avec CQ = 0.1	- **Si** ($Satisfaction_{proximité} > 7$) **et** ($Satisfaction_{densité} > 7$) **et** ($Satisfaction_{satisfactionBat} > 5$) **alors** *état optimal* - **Sinon** *état non optimal*
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	état toujours *non optimal*
	Révision avec CQ = 0	- **Si** ($Satisfaction_{BatimentsCoins} \leq 6$) **alors** *état optimal* - **Sinon** *état non optimal*
	Révision avec CQ = 0.01	- **Si** ($Satisfaction_{BatimentsCoins} \leq 6$) **alors** *état optimal* - **Sinon** *état non optimal*
	Révision avec CQ = 0.1	état toujours *non optimal*

Nous pouvons constater que l'ensemble des règles apprises concernent la satisfaction de la contrainte de répartition des bâtiments dans les coins. Nous pouvons expliquer cette règle par le fait que la contrainte de répartition des bâtiments dans les coins est une contrainte de conservation qui ne propose pas d'action. Lorsque cette contrainte a une valeur faible de satisfaction, il n'est a priori pas possible de l'améliorer (il n'y a pas d'action chargée de replacer des bâtiments dans les coins).

Les résultats (figure F.25, F.26 et F.27) obtenus avec les jeux de connaissances révisés avec un *coefficient de qualité* nul montrent une légère amélioration de l'efficience du système de généralisation pour les trois jeux de connaissances initiaux sans affectation visible de l'efficacité du système.

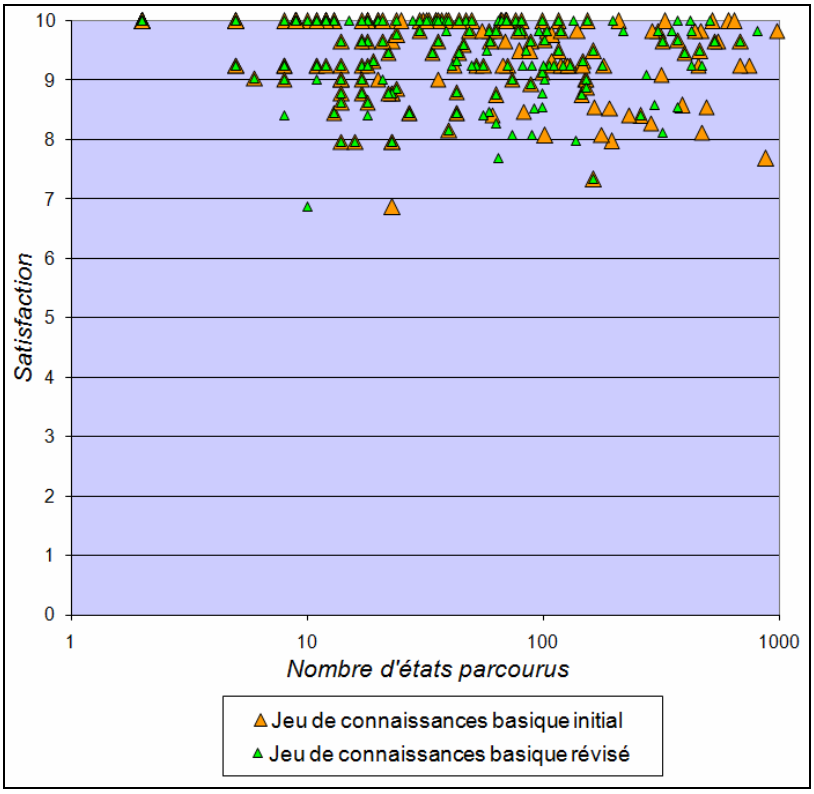


Figure F.25 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) de la connaissance relative à l'optimalité des états (échelle logarithmique pour le nombre d'états parcourus)

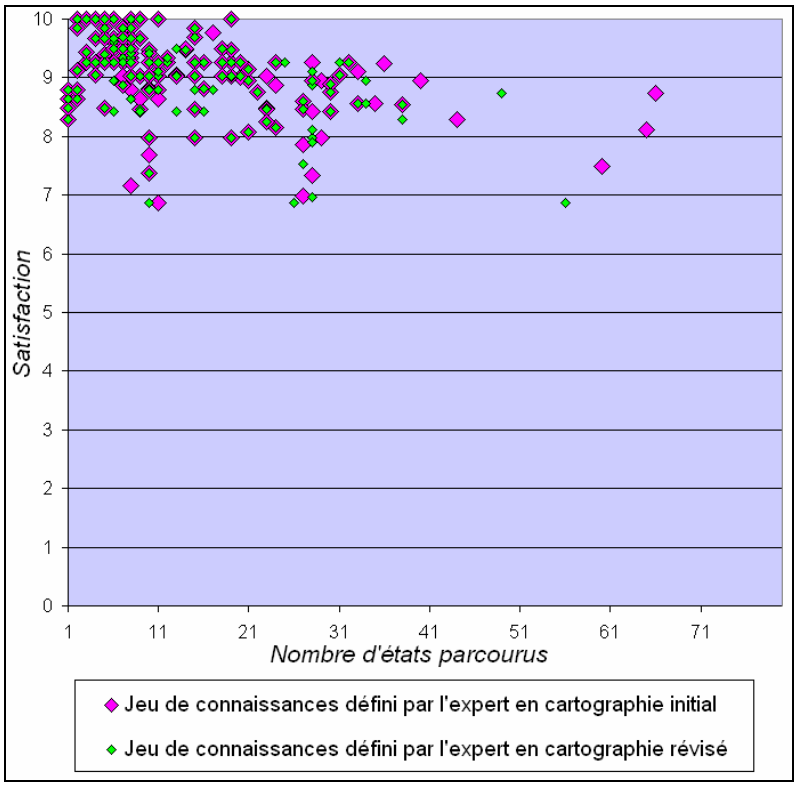


Figure F.26 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (CQ = 0) de la connaissance relative à l'optimalité des états

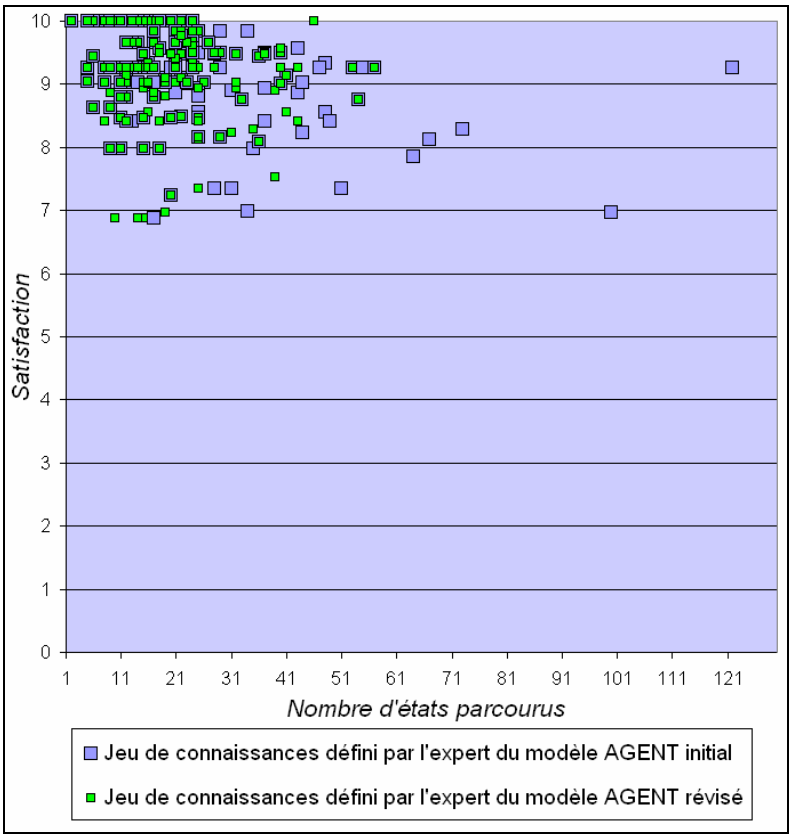


Figure F.27 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (CQ = 0) de la connaissance relative à l'optimalité des états

Concernant les résultats cartographiques obtenus sur la zone de test avant et après révision des connaissances (avec un *coefficient de qualité de la connaissance* de 0), nous observons figures F.28, F.29 et F.30 que les jeux de connaissances révisés permettent d'obtenir des résultats très similaires aux résultats initiaux. Ainsi, il n'y a pas de différences apparentes sur la figure F.28 entre les résultats obtenus avant et après révision par le jeu de connaissances *basique*. Concernant les deux autres jeux de connaissances (figures F.29 et F.30), on peut noter l'apparition de quelques problèmes de proximité et de densité ; la disparition de certains problèmes de bâtiments non généralisés pour le jeu de connaissances *défini par l'expert en cartographie* est observée.

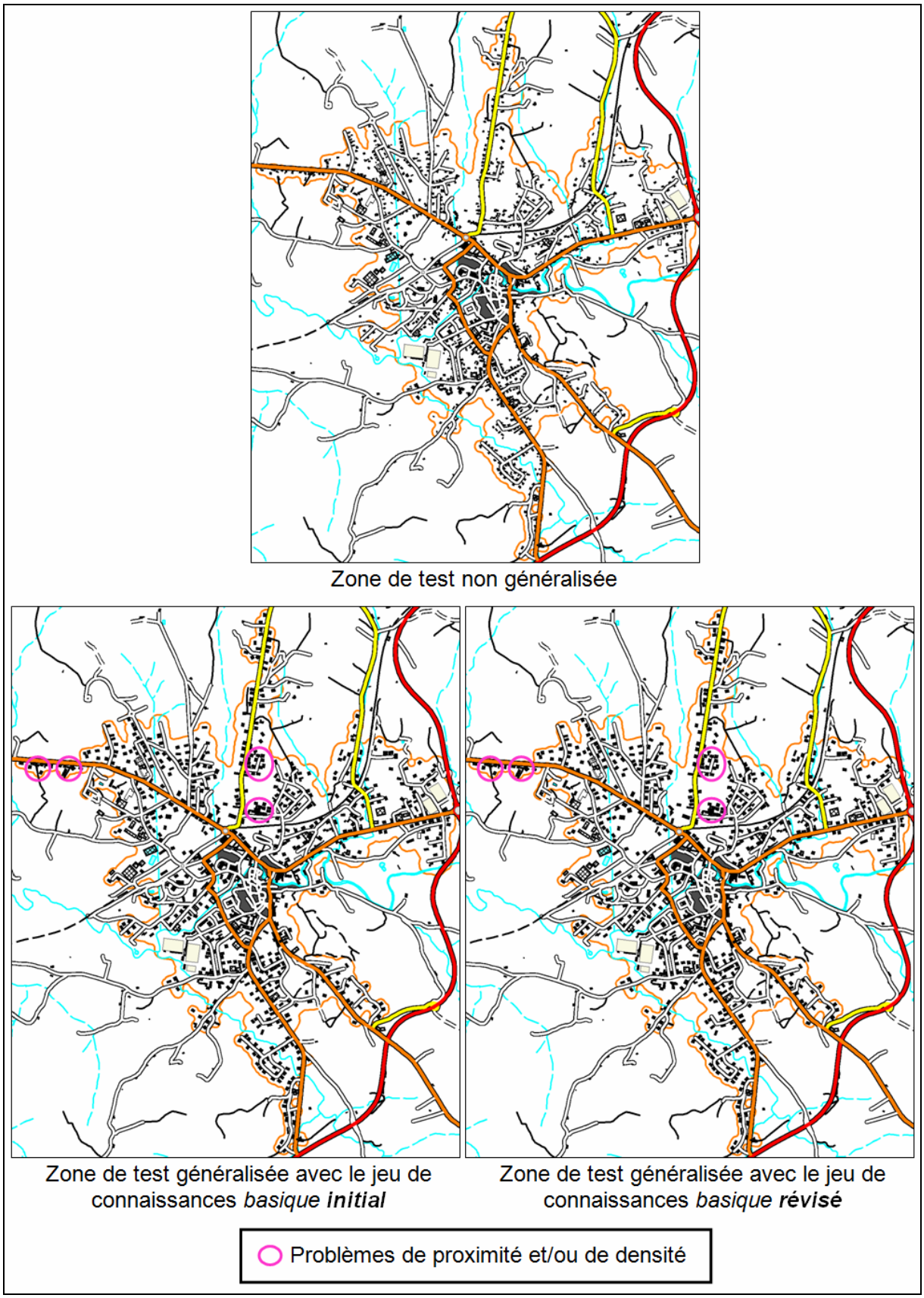


Figure F.28 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) de la connaissance relative à l'optimalité des états

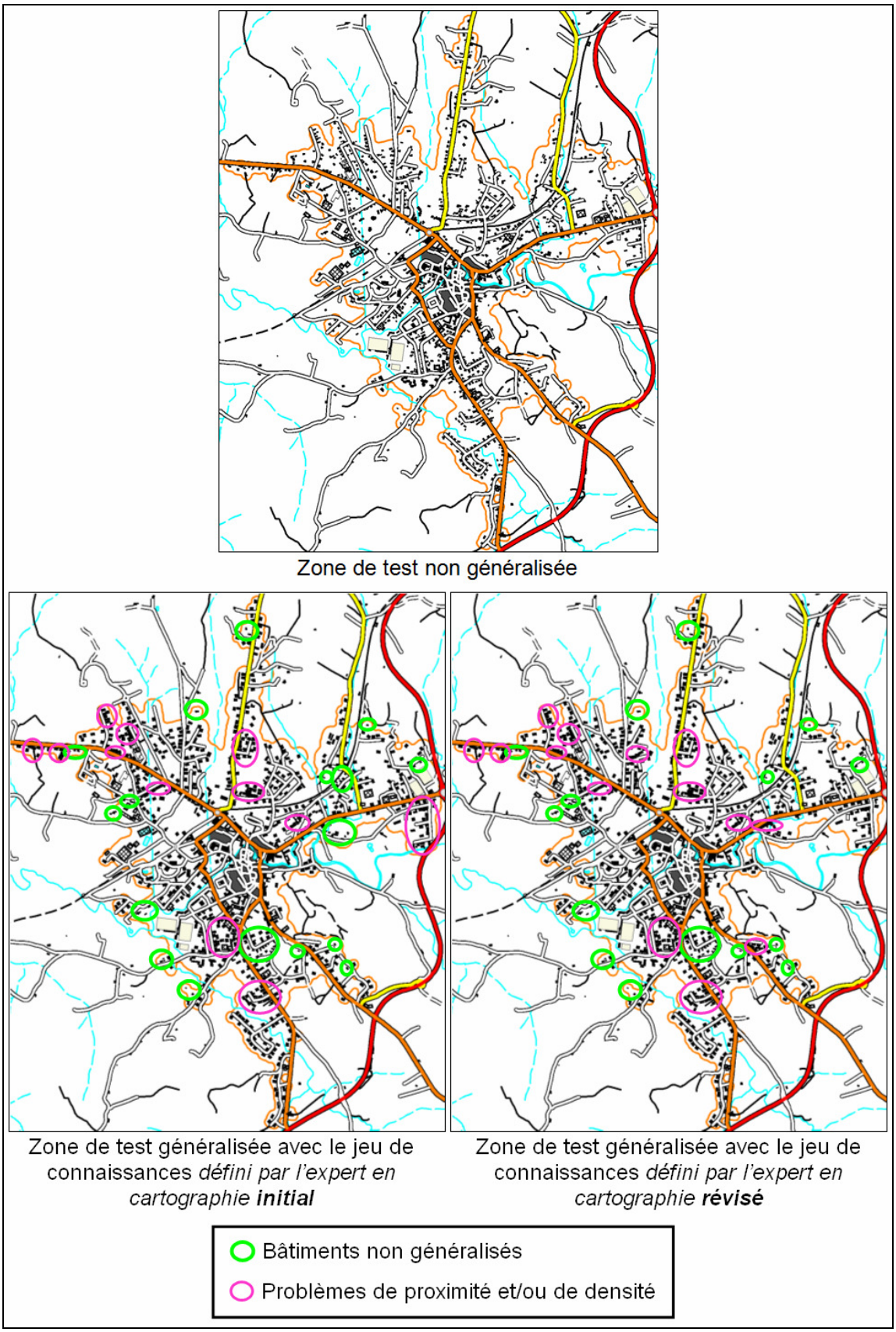


Figure F.29 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) de la connaissance relative à l'optimalité des états

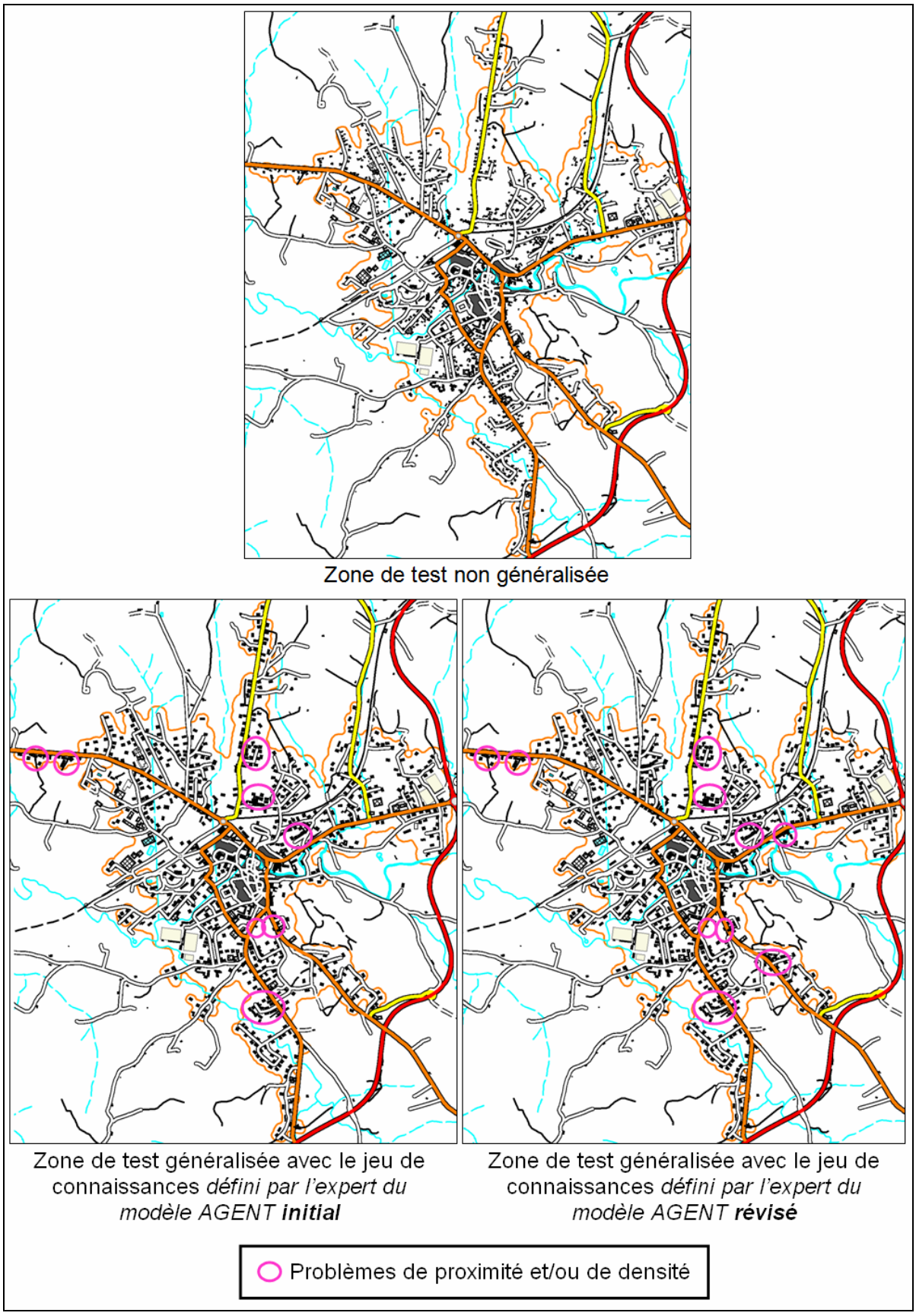


Figure F.30 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (avec CQ = 0) de la connaissance relative à l'optimalité des états

Les tables suivantes donnent les résultats des jeux de connaissances obtenus après révision sur la zone de test.

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.862**	0.355	0.761
Après révision avec ***CQ*** *= 0*	0.857	**0.386**	**0.763**
Après révision avec ***CQ*** *= 0.01*	0.857	**0.386**	**0.763**
Après révision avec ***CQ*** *= 0.1*	**0.862**	0.355	0.761

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.807**	0.69	0.784
Après révision avec ***CQ*** *= 0*	0.806	**0.708**	**0.786**
Après révision avec ***CQ*** *= 0.01*	**0.807**	0.69	0.784
Après révision avec ***CQ*** *= 0.1*	**0.807**	0.69	0.784

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.838**	0.599	0.79
Après révision avec ***CQ*** *= 0*	0.832	**0.638**	**0.793**
Après révision avec ***CQ*** *= 0.01*	0.832	**0.638**	**0.793**
Après révision avec ***CQ*** *= 0.1*	**0.838**	0.599	0.79

Ces résultats confirment que la révision des connaissances d'optimalité a permis, au prix d'une légère baisse d'efficacité du système, un gain d'efficience. Le processus de révision a permis, pour les trois jeux de connaissances, une légère amélioration de la performance du système de généralisation.

Expérimentation de l'approche de simplification des bases de règles

Nous nous intéressons ici à la validation de notre approche de simplification des bases de règles. Nous proposons de tester les jeux de connaissances obtenus après révision des jeux de connaissances initiaux avec et sans simplification des bases de règles. Afin de biaiser le moins possible les résultats par les connaissances initiales, nous proposons de prendre pour ces tests un *coefficient de qualité de la connaissance* égale à 0.

Les tables suivantes donnent les résultats obtenus sur la zone de test.

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**	***Nombre de règles***
Avant révision	**0.862**	0.355	0.761	0
Avec phase de simplification de la base de règles	0.857	**0.386**	**0.763**	7
Sans phase de simplification de la base de règles	0.857	**0.386**	**0.763**	1

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**	***Nombre de règles***
Avant révision	**0.838**	0.599	**0.79**	0
Sans phase de simplification de la base de règles	0.832	**0.638**	0.793	12
Avec phase de simplification de la base de règles	0.832	**0.638**	0.793	1

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**	***Nombre de règles***
Avant révision	**0.807**	0.69	0.784	0
Sans phase de simplification de la base de règles	**0.807**	0.69	0.784	7
Avec phase de simplification de la base de règles	0.806	**0.708**	**0.786**	1

Nous constatons que la phase de simplification de la base de règles a bien permis de diminuer de manière conséquente le nombre de règles (et donc la lisibilité de la base de règles) sans perte de performance, entraînant même une amélioration des performances pour le jeu de connaissances *défini par l'expert du modèle AGENT*. Il est possible d'expliquer cette amélioration par le fait que la simplification peut permettre de faire disparaître (par agrégation) des règles spécifiques à l'échantillon d'instances utilisé pour le partitionnement et l'affectation des conclusions. En effet, nous rappelons que la simplification est basée sur l'utilisation d'un échantillon d'instances différent de celui utilisé pour le partitionnement et l'affectation des conclusions. Cette expérimentation démontre l'intérêt de la phase de simplification de la base de règles.

Comparaison des approches de partitionnement

Nous avons testé les jeux de connaissances obtenus après révision des jeux de connaissances initiaux avec utilisation des deux approches de partitionnement (et *CQ = 0*). Les tables suivantes donnent les résultats obtenus.

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**	***Nombre de partitions crées***
Avant révision	**0.862**	0.355	0.761	-
Avec approche de partitionnement par apprentissage/comparaison	0.857	**0.386**	**0.763**	7
Avec approche de partitionnement par discrétisation des mesures	0.856	0.385	0.762	192

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**	***Nombre de partitions crées***
Avant révision	0.807	0.69	0.784	-
Avec approche de partitionnement par apprentissage/comparaison	0.806	**0.708**	**0.786**	128
Avec approche de partitionnement par discrétisation des mesures	**0.808**	0.699	**0.786**	128

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**	***Nombre de partitions crées***
Avant révision	**0.838**	0.599	0.79	-
Avec approche de partitionnement par apprentissage/comparaison	0.832	**0.638**	**0.793**	4
Avec approche de partitionnement par discrétisation des mesures	0.837	0.61	0.791	12

Nous constatons que les deux approches donnent des résultats proches. Les résultats obtenus avec l'approche par apprentissage et comparaison de règles sont légèrement meilleurs à la fois d'un point de vue des performances mais également du point de vue du nombre de partitions générées. L'intérêt de générer moins de partitions est de faciliter ensuite la phase d'exploration. En effet, moins de partitions signifie une combinatoire moins importante pour la recherche des meilleures conclusions à affecter. Un défaut de l'approche par apprentissage et comparaison de règles reste néanmoins sa complexité algorithmique (cf. D.IV.3.4.3) : le temps nécessaire au partitionnement de l'espace des règles devient handicapant au-delà d'une quinzaine de règles apprises, dépendantes chacune de cinq ou six mesures différentes. Il est indispensable de choisir pour cette approche un algorithme d'apprentissage ayant un fort potentiel de généralisation.

F.III.4.3 Bilan de l'expérimentation

Cette expérimentation a permis de valider l'application de notre approche de révision pour la connaissance relative à l'optimalité des états. Nous avons en effet pu constater une amélioration des trois jeux de connaissances par la révision de leurs règles d'optimalité des états.

Cette expérimentation a également mis en lumière l'intérêt de l'intégration d'une phase de simplification des bases de règles dans le processus de révision : les bases de règles obtenues sont ainsi plus simples (dont plus lisibles) et peuvent même s'avérer plus performantes.

Nous avons enfin testé nos deux approches de partitionnement de l'espace des mesures. Nous avons constaté que ces deux approches permettaient bien d'améliorer les jeux de connaissances. L'approche par apprentissage et comparaison de règles s'est avérée plus efficace pour cette expérimentation mais peut poser des problèmes de complexité algorithmique lorsque trop de règles sont apprises pour partitionner l'espace des mesures.

F.III.5 Révision de la connaissance relative à la fin de cycle

F.III.5.1 Contexte de l'expérimentation

Nous avons donné la possibilité dans notre modèle enrichi de définir des règles de fin de cycle (cf. B.V.3.1.1). La connaissance relative à la fin de cycle est représentée sous la forme d'une base de règles de $\mathcal{BR}$ (cf. D.IV.3.2).

Nous appliquons pour cette connaissance la même approche que celle utilisée pour la connaissance relative à l'optimalité des états.

Nous testons également nos deux approches de partitionnement et nous utilisons, comme algorithme d'exploration de l'espace des affectations possibles de conclusion, l'algorithme de recherche locale taboue.

Comme la connaissance relative à l'optimalité des états, cette connaissance a un rôle d'élagage. Sa révision ne pourra donc pas permettre de gain d'efficacité pour le jeu de connaissances *basique* et celui *défini par l'expert du modèle AGENT* pour lesquels aucune règle de fin de cycle n'est définie. Le seul jeu de connaissances pour lequel la révision de cette connaissance a une chance d'améliorer l'efficacité est le jeu de connaissances *défini par l'expert en cartographie* qui comporte une règle de fin de cycle.

F.III.5.2 Résultats obtenus

Le tableau suivant présente les règles de fin de cycle obtenues après révision avec l'approche de partitionnement par apprentissage et comparaison de règles.

		Règles de fin de cycle
Jeu de connaissances *basique*	Avant révision	Pas de règles de fin de cycle
	Révision avec CQ = 0	- **Si** min_satisfactionBat > 6.77 **et** min_satisfactionBat ≤ 9.41 **et** $satisfaction_{proximité}$ > 5 **et** $satisfaction_{proximité}$ ≤ 9 **et** $satisfaction_{répartitionSpat}$ > 8, **alors** *arrêt* - **Si** min_satisfactionBat > 6.77 **et** $satisfaction_{proximité}$ > 4 **et** $satisfaction_{proximité}$ ≤ 5 **et** $satisfaction_{répartitionSpat}$ ≤ 7, **alors** *arrêt* - **Si** min_satisfactionBat > 9.41 **et** $satisfaction_{proximité}$ > 6 **et** $satisfaction_{proximité}$ ≤ 9 **et** $satisfaction_{répartitionSpat}$ > 7, **alors** *arrêt*
	Révision avec CQ = 0.01	- **Si** min_satisfactionBat > 6.77 **et** min_satisfactionBat ≤ 9.41 **et** $satisfaction_{proximité}$ > 5 **et** $satisfaction_{proximité}$ ≤ 9 **et** $satisfaction_{répartitionSpat}$ > 8, **alors** *arrêt* - **Si** min_satisfactionBat > 6.77 **et** $satisfaction_{proximité}$ > 4 **et** $satisfaction_{proximité}$ ≤ 5 **et** $satisfaction_{répartitionSpat}$ ≤ 7, **alors** *arrêt* - **Si** min_satisfactionBat > 9.41 **et** $satisfaction_{proximité}$ > 6 **et** $satisfaction_{proximité}$ ≤ 9 **et** $satisfaction_{répartitionSpat}$ > 7, **alors** *arrêt*
	Révision avec CQ = 0.1	Pas de règles de fin de cycle
Jeu de connaissances *défini par l'expert en cartographie*	Avant révision	- **Si** $satisfaction_{satisfactionBat}$ > 7 **et** $satisfaction_{densité}$ > 5 **et** $satisfaction_{proximité}$ > 7 **et** $satisfaction_{répartitionSpat}$ > 7, **alors** *arrêt*
	Révision avec CQ = 0	Pas de règles de fin de cycle
	Révision avec CQ = 0.01	Pas de règles de fin de cycle
	Révision avec CQ = 0.1	Pas de règles de fin de cycle
Jeu de connaissances *défini par l'expert du modèle AGENT*	Avant révision	Pas de règles de fin de cycle
	Révision avec CQ = 0	Pas de règles de fin de cycle
	Révision avec CQ = 0.01	Pas de règles de fin de cycle
	Révision avec CQ = 0.1	Pas de règles de fin de cycle

Nous constatons que le processus de révision n'a pas permis de définir de règles pertinentes pour deux des jeux de connaissances. Ainsi, le seul jeu de connaissances pour lequel des règles de fin de cycle ont été définies est le jeu de connaissances *basique*. Le fait que des règles de fin de cycle aient été définies pour ce jeu de connaissances peut s'expliquer par le fait que le score d'efficience obtenu par le jeu de connaissances *basique* est mauvais contrairement au score obtenu par les deux autres jeux de connaissances. Or, l'ajout de règles de fin de cycle ne peut qu'améliorer l'efficience du système et non son efficacité. Il est donc plus simple de définir une règle de fin de cycle permettant d'améliorer de manière conséquente le score d'efficience (de façon à compenser la perte d'efficacité) pour le jeu de connaissances *basique* que pour les deux autres jeux de connaissances.
Il est également possible de constater que la règle de fin de cycle *définie par l'expert en cartographie* n'a pas été conservée par le processus de révision.

Les résultats (figure F.31) obtenus avec le jeu de connaissances *basique* révisé avec un *coefficient de qualité* nul montrent une amélioration sensible de l'efficience du système de généralisation sans détérioration importante de son efficacité.

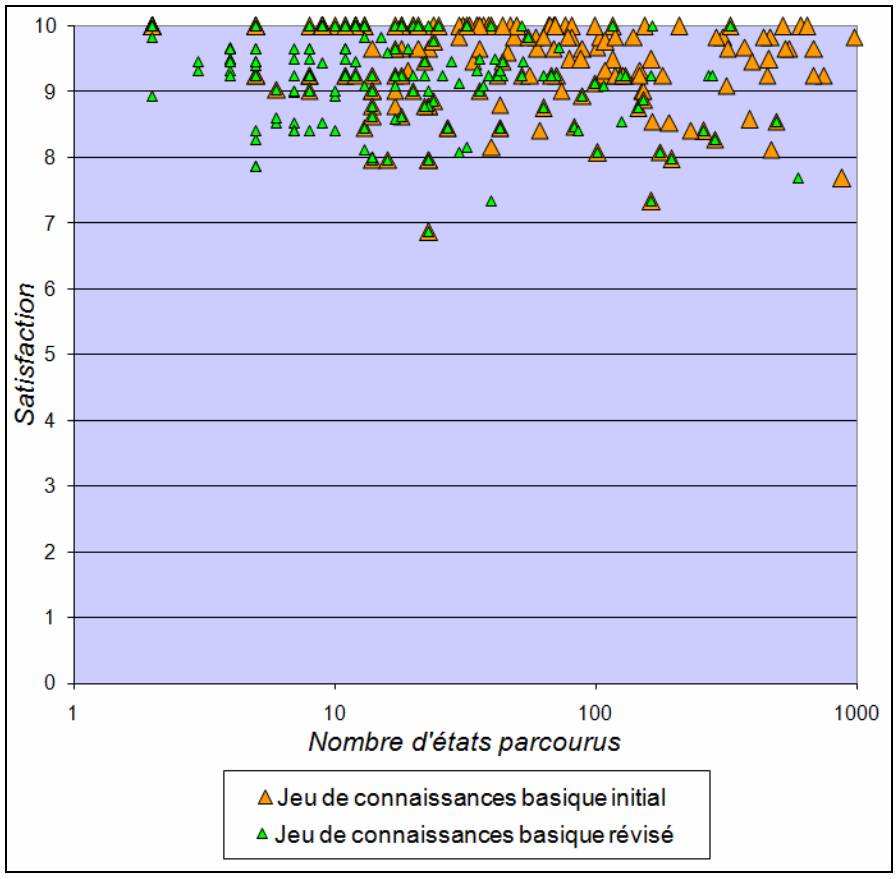


Figure F.31 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) de la connaissance relative à la fin de cycle (échelle logarithmique pour le nombre d'états parcourus)

Concernant les résultats obtenus avec le jeu de connaissances *défini par l'expert en cartographie* révisé avec un *coefficient de qualité* nul (figure F.32), ces derniers montrent une vraie amélioration de l'efficacité du système de généralisation (davantage de *groupements de*

bâtiments ont une valeur de satisfaction de 10 ou proche de 10) sans détérioration importante de son efficience.

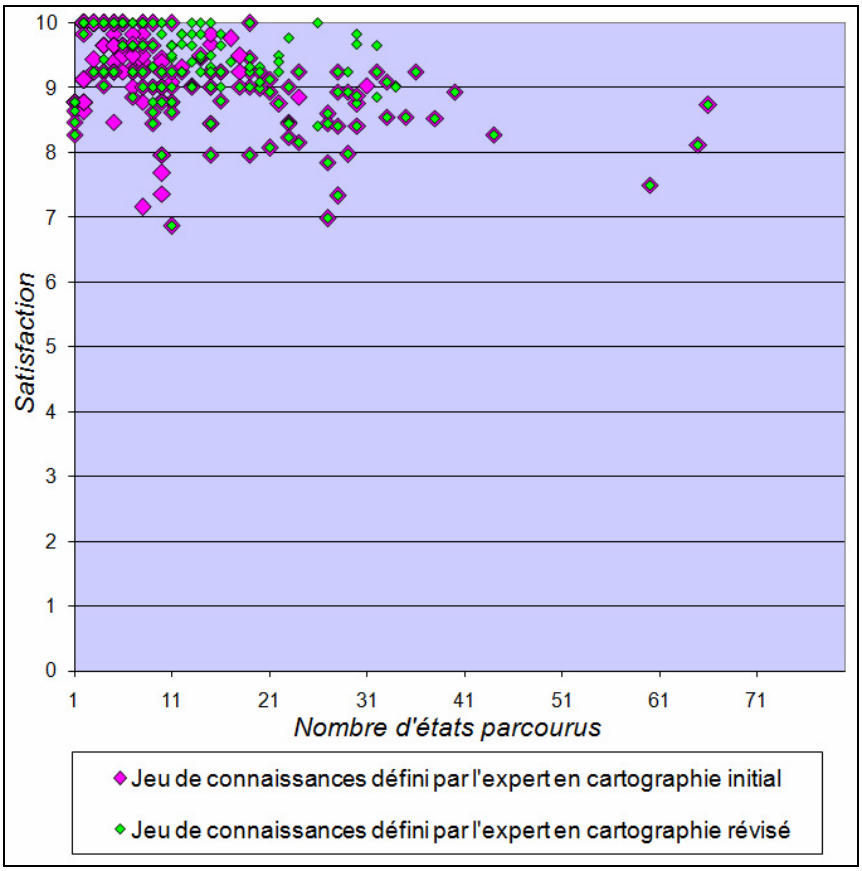


Figure F.32 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) de la connaissance relative à la fin de cycle

Concernant les résultats cartographiques obtenus sur la zone de test avant et après révision (avec un *coefficient de qualité de la connaissance* de 0) du jeu de connaissances *basique* (figure F.33), nous observons que certains problèmes de proximité et de densité ont disparu. Néanmoins, la répartition spatiale des bâtiments est dans l'ensemble moins bonne avec le jeu révisé et nous pouvons observer des suppressions de bâtiments importants. Dans leur globalité, les résultats obtenus avec le jeu de connaissances révisé sont donc légèrement moins bons.
Pour les résultats cartographiques obtenus sur la zone de test avant et après révision (avec un *coefficient de qualité de la connaissance* de 0) du jeu de connaissances *défini par l'expert en cartographie* (figure F.34), nous observons que certains problèmes de proximité et de densité ont disparu ainsi que certains problèmes de bâtiments non généralisés. Les résultats obtenus avec le jeu de connaissances révisé sont dans l'ensemble sensiblement meilleurs.

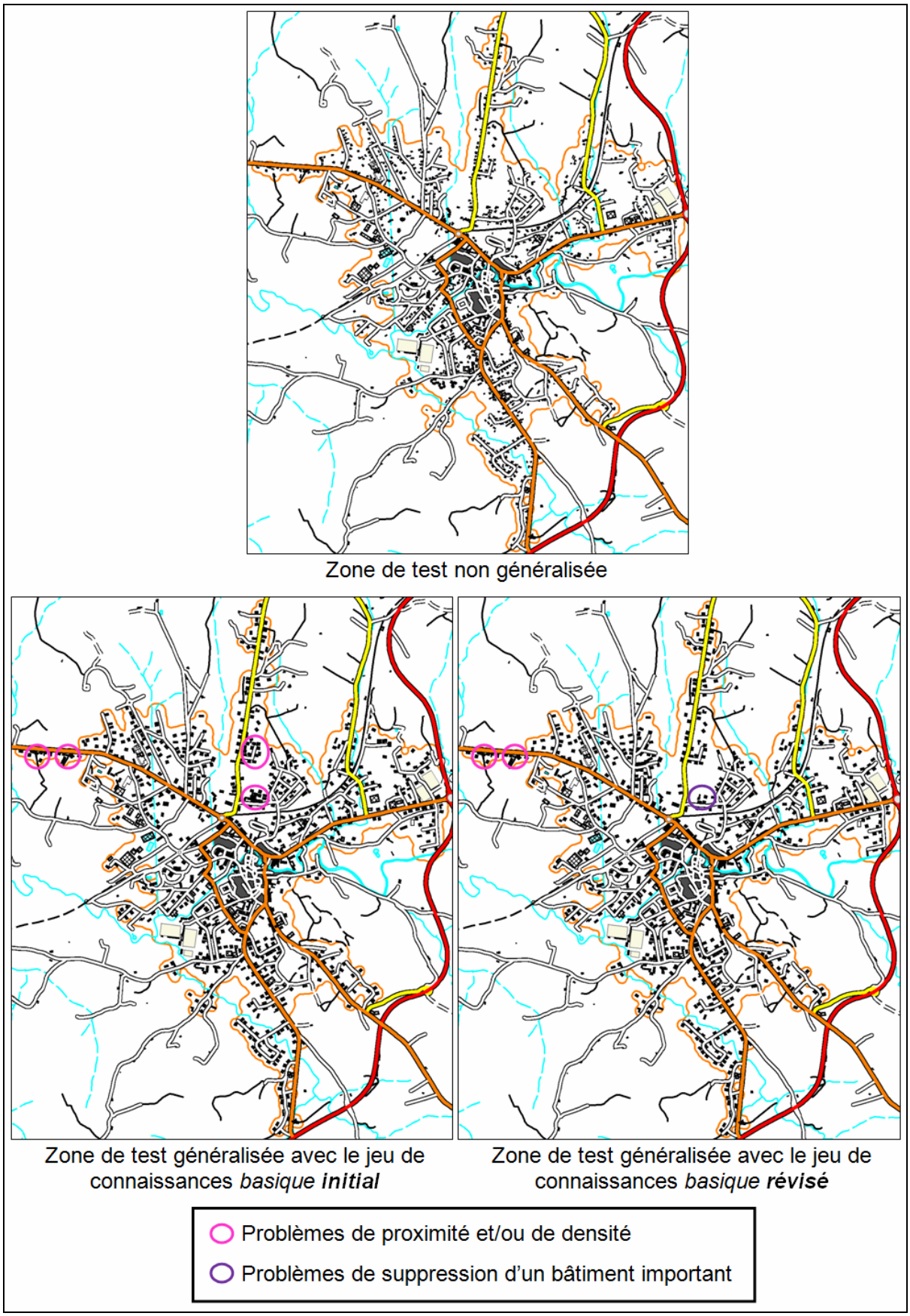


Figure F.33 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) de la connaissance relative à la fin de cycle

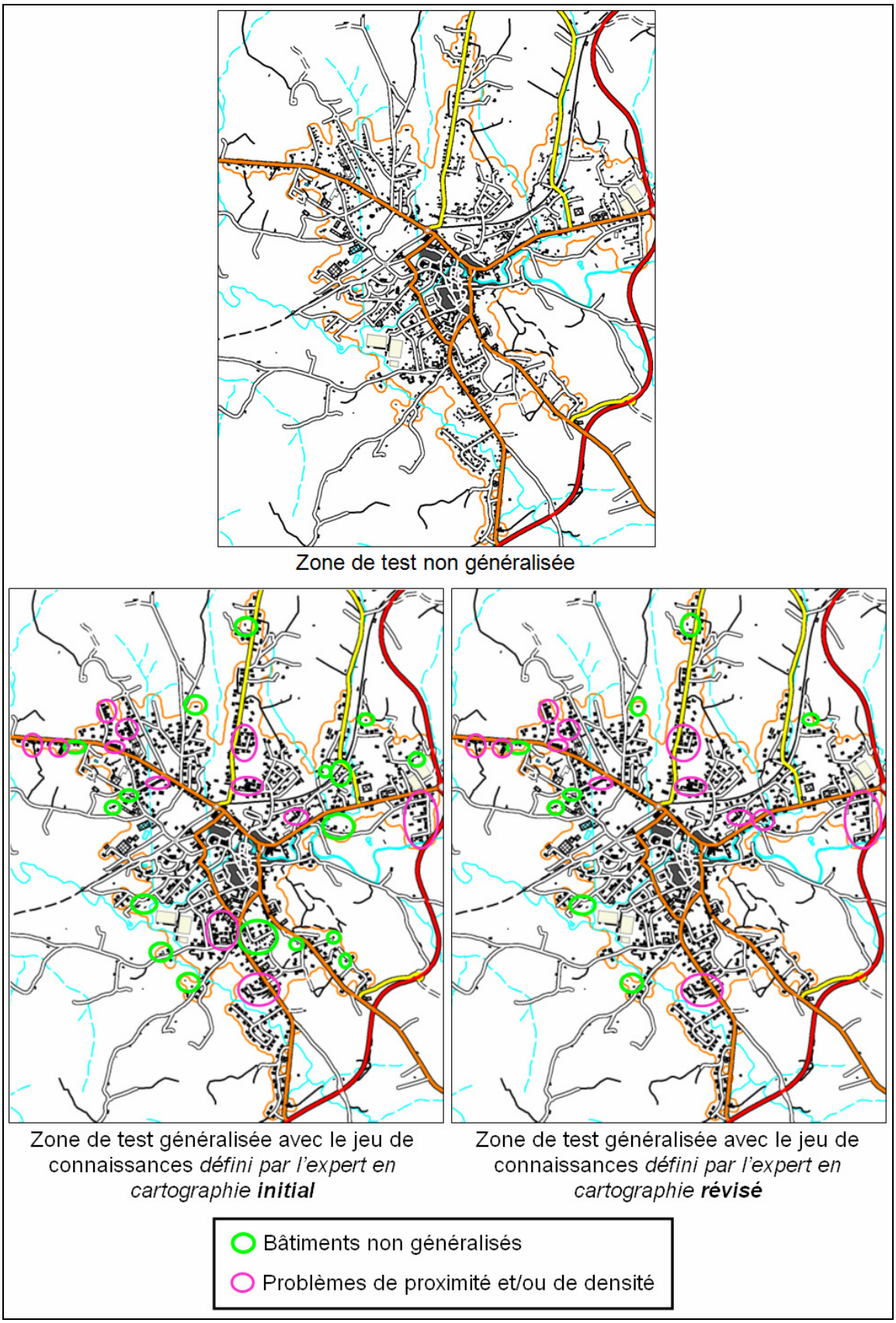


Figure F.34 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) de la connaissance relative à la fin de cycle

Les tables suivantes donnent les résultats des jeux de connaissances obtenus après révision sur la zone de test en utilisant la méthode de partitionnement par apprentissage et comparaison de règles.

Révision de la connaissance relative à la fin de cycle pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.862**	0.355	0.761
Après révision avec ***CQ*** *= 0*	0.83	**0.522**	**0.768**
Après révision avec ***CQ*** *= 0.01*	0.83	**0.522**	**0.768**
Après révision avec ***CQ*** *= 0.1*	**0.862**	0.355	0.761

Révision de la connaissance relative à la fin de cycle pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	0.807	0.69	0.784
Après révision avec ***CQ*** *= 0*	**0.824**	**0.655**	**0.789**
Après révision avec ***CQ*** *= 0.01*	**0.824**	**0.655**	**0.789**
Après révision avec ***CQ*** *= 0.1*	**0.824**	**0.655**	**0.789**

Nous constatons que la révision de la connaissance relative à la fin de cycle a bien permis d'améliorer les performances de deux jeux de connaissances. Ainsi, l'ajout de règles de fin de cycle a permis au jeu de connaissances *basique* d'améliorer significativement l'efficience obtenue par le système de généralisation avec ce jeu de connaissances entraînant une hausse de ses performances (grâce à une baisse suffisamment faible de son efficacité). Concernant le jeu de connaissances *défini par l'expert en cartographie*, la suppression de la règle de fin de cycle a permis au système de généralisation d'améliorer son score d'efficacité et ainsi ses performances.

Comparaison des approches de partitionnement

Nous avons testé les jeux de connaissances obtenus après révision des jeux de connaissances initiaux avec utilisation des deux approches de partitionnement (avec *CQ = 0*). Les tables suivantes donnent les résultats obtenus.

Révision de la connaissance relative à l'optimalité des états pour le jeu de connaissances *basique* :

	Efficacité(S_K,P_n)	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***	***Nombre de partitions créées***
Avant révision	**0.862**	0.355	0.761	-
Avec approche de partitionnement par apprentissage/comparaison	0.83	0.522	0.768	7
Avec approche de partitionnement par discrétisation des mesures	0.811	**0.606**	**0.77**	256

Révision de la connaissance relative à la fin de cycle pour le jeu de connaissances *défini par l'expert en cartographie* :

	Efficacité(S_K,P_n)	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***	***Nombre de partitions créées***
Avant révision	0.807	**0.69**	0.784	-
Avec approche de partitionnement par apprentissage/comparaison	**0.824**	0.655	0.789	5
Avec approche de partitionnement par discrétisation des mesures	0.822	0.665	**0.79**	8

Révision de la connaissance relative à la fin de cycle pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	Efficacité(S_K,P_n)	***Efficience(S_K,P_n)***	***Perf(S_K,P_n)***	***Nombre de partitions créées***
Avant révision	**0.838**	**0.599**	**0.79**	-
Avec approche de partitionnement par apprentissage/comparaison	**0.838**	**0.599**	**0.79**	7
Avec approche de partitionnement par discrétisation des mesures	0.836	0.606	**0.79**	16

Nous constatons que les deux approches donnent des résultats très proches. Dans le cadre de cette expérimentation, les résultats obtenus avec l'approche par discrétisation indépendante des mesures sont légèrement meilleurs du point de vue des performances.

F.III.5.3 Bilan de l'expérimentation

Cette expérimentation a permis de valider l'application de notre approche de révision pour la révision de la connaissance relative à la fin de cycle. Notre approche a permis d'améliorer deux des trois jeux de connaissances (il n'y a pas eu de changements pour le jeu de connaissances *défini par l'expert du modèle AGENT*).

Nous avons pu tester à nouveau nos deux approches de partitionnement de l'espace des mesures. Nous avons constaté que ces deux approches ont permis d'améliorer les performances des jeux de connaissances. L'approche par discrétisation de mesures s'est avérée légèrement plus performante que l'approche par apprentissage et comparaison de règles pour cette expérimentation.

F.III.6 Révision des connaissances relatives à la priorité des contraintes

F.III.6.1 Contexte de l'expérimentation

La priorité d'une contrainte traduit l'urgence de traitement de cette contrainte (cf. B.V.3.1.1). La connaissance relative à la priorité d'une contrainte est représentée sous la forme d'une base de règles de $\mathcal{BR}$ (cf. D.IV.3.2). Chaque contrainte dispose d'une telle base de règles définissant sa priorité en fonction de sa satisfaction.

Nous avons proposé d'utiliser pour ce type de connaissances la même approche que celle des connaissances relatives à l'optimalité des états et à la fin de cycle : dans une première étape, nous partitionnons l'espace des mesures en zones admettant une conclusion a priori homogène, puis dans une seconde étape, nous cherchons à affecter la meilleure conclusion possible à chaque zone, et enfin dans une dernière étape nous simplifions les bases de règles obtenues.

Nous n'utilisons pas pour ces connaissances de priorité les approches de partitionnement que nous avons testées durant nos deux précédentes expérimentations. En effet, étant donné que les règles de priorité de chaque contrainte dépendent seulement de la satisfaction de la contrainte qui a pour valeur un entier compris entre 1 et 10, nous proposons de partitionner l'espace des mesures en dix zones représentant chacune une valeur de satisfaction.
La priorité des contraintes étant exprimée sous la forme d'un entier compris entre 1 et 5, le nombre de bases de règles possibles pour une contrainte est de 10^5.

En raison des interdépendances directes existantes entre les connaissances relatives aux priorités des différentes contraintes, il est nécessaire de réviser l'ensemble des connaissances relatives aux priorités des contraintes en même temps. La taille de l'espace de recherche est donc de $10^{5\times\text{nombre de contraintes}}$.

Nous proposons pour ce type de connaissances de modifier légèrement la fonction de performance définie en partie F.II.3.3. En effet, ces connaissances n'interviennent pas dans l'élagage des arbres d'états, mais seulement dans l'ordre d'application des actions (cf. C.III.2). La révision des règles de priorité des contraintes a pour enjeu de permettre au système de converger plus vite vers le meilleur état (de le mettre le plus à gauche possible dans l'arbre). Ainsi, pour un jeu de connaissances dans lequel le critère de validité ne dépend pas de l'ensemble des états parcourus mais juste de l'état précédent (***CEPAC***, ***CEPAC ET CEPAG***), l'efficacité du processus ne pourra jamais être modifiée par la révision des règles de

priorité des contraintes. De même l'efficience ne sera modifiée que pour les *groupements de bâtiments* pour lesquels il existe un état parfait.
Nous proposons donc de ne pas prendre en compte le nombre total d'états parcourus mais le nombre d'états parcourus pour arriver au meilleur état. Notons que ces deux nombres peuvent être égaux dans le cas où un état parfait a été trouvé du premier coup. Nous ne proposons pas ici de résultats cartographiques car les connaissances relatives à la priorité des contraintes ont avant tout pour rôle de permettre au système de converger plus rapidement vers le meilleur état.

Nous allons particulièrement nous intéresser dans le cadre de cette expérimentation à l'étape d'exploration de l'espace des affectations possibles de conclusion. Ainsi, nous avons proposé au chapitre D deux approches d'exploration pour ce type de connaissances.

La première est celle que nous avons utilisée lors des deux précédentes explorations : l'utilisation d'une métaheuristique de recherche locale (une recherche locale taboue). Nous avons défini pour les connaissances admettant pour conclusion un ensemble de valeurs ordonnées (comme c'est le cas pour les priorités des contraintes) un voisinage dépendant d'un paramètre *k* traduisant la « taille » du voisinage (cf. D.IV.3.5.4.2). La priorité d'une contrainte étant représentée par un entier compris entre 1 et 5, l'écart entre la valeur minimale et la valeur maximale est 4. Nous proposons de prendre en considération la plus petite valeur k (k = 1) et la plus grande valeur de k (k = 4). Nous testons au travers d'une expérimentation deux valeurs pour *k* (1 et 4). Un voisinage plus large (k = 4) rend la phase de sélection du meilleur mouvement plus longue (plus de voisin à tester) mais permet de choisir à chaque itération un meilleur mouvement (plus de choix pour le mouvement).

La seconde approche que nous avons proposée est une approche agent basée sur une analyse locale au niveau de chaque partition (cf. D.IV.3.5.5). Cette approche demande de paramétrer un coefficient de filtrage (*CF*). Nous rappelons que ce coefficient est compris entre 0 et 1. Le 0 signifiant l'absence de filtrage et le 1, un filtrage complet. Nous avons défini empiriquement trois valeurs différentes (0.25, 0.5, 0.75) que nous testerons.

Nous avons enfin proposé de combiner notre approche agent à une métaheuristique de recherche locale classique (cf. D.IV.3.5.5.5). Nous proposons ainsi de combiner notre approche agent à une recherche locale taboue. Comme nous l'avons évoqué en partie (cf. D.IV.3.5.5.5), l'objectif du filtrage est de diminuer le nombre de voisins considérés à chaque itération. Nous proposons donc de prendre en compte pour la recherche locale taboue le plus grand voisinage possible et donc de prendre une valeur pour k égale à 4.
Cette approche demande également de paramétrer un coefficient de filtrage (*CF*). Nous avons défini empiriquement deux valeurs pour le coefficient de filtrage que nous testons : 0.1 et 0.5.

F.III.6.2 Résultats obtenus

Comparaison des approches d'exploration

Nous nous intéressons dans un premier temps aux résultats obtenus lors de l'exploration de l'espace des affectations possibles de conclusion. Toutes les expérimentations ont été effectuées avec un *coefficient de qualité des connaissances* nul.

Les figures F.35, F.36 et F.37 montrent les résultats obtenus avec les deux valeurs de *k* testées. Nous remarquons sur ces figures que les résultats sont très différents en fonction des jeux de connaissances. Pour le jeu de connaissances *basique*, la valeur de k égale à 1 a permis

au processus d'exploration de converger plus vite vers la meilleure solution. Concernant le jeu de connaissances *défini par l'expert en cartographie*, la valeur de k égale à 4 a permis au processus d'exploration de trouver une meilleure solution. Inversement, pour le jeu de connaissances *défini par l'expert du modèle AGENT*, c'est la valeur de k égale à 1 qui a permis de trouver une meilleure solution mais la convergence fut nettement plus lente qu'avec la valeur de k égale à 4. Nous pouvons expliquer cela par le fait que pour le jeu de connaissances *basique*, l'obtention de la meilleure solution a dû nécessiter de modifier légèrement la valeur de conclusion de nombreuses partitions. Ainsi, la recherche menée avec la valeur de *k* égale à 1 a permis d'obtenir plus rapidement cette meilleure solution. Au contraire, pour le jeu de connaissances *défini par l'expert du modèle AGENT*, l'obtention de la meilleure solution a dû nécessiter de modifier de façon importante la valeur de conclusion de seulement quelques partitions et donc de permettre à la recherche menée avec la valeur de *k* égale à 4 de converger plus rapidement vers la meilleure solution.

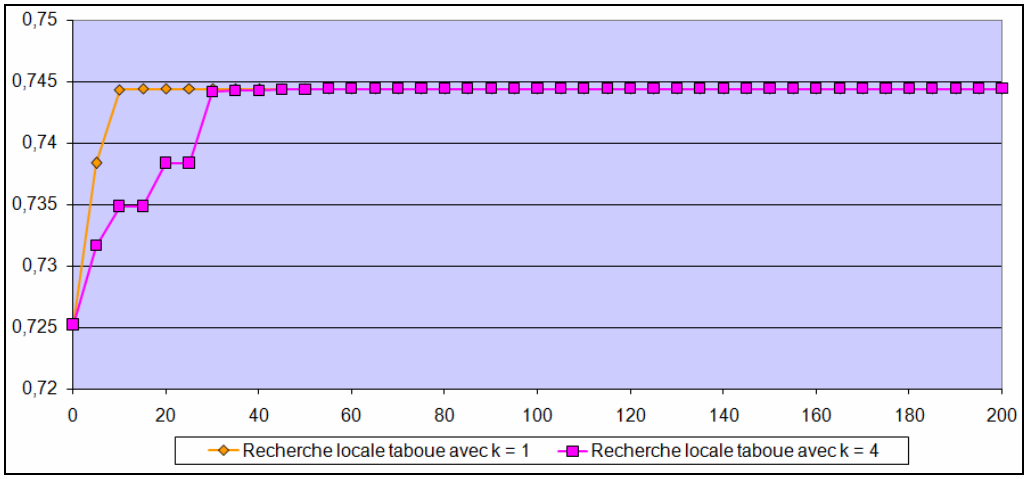


Figure F.35 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *basique* pour la recherche locale taboue (avec *k = 1* et *k = 4*)

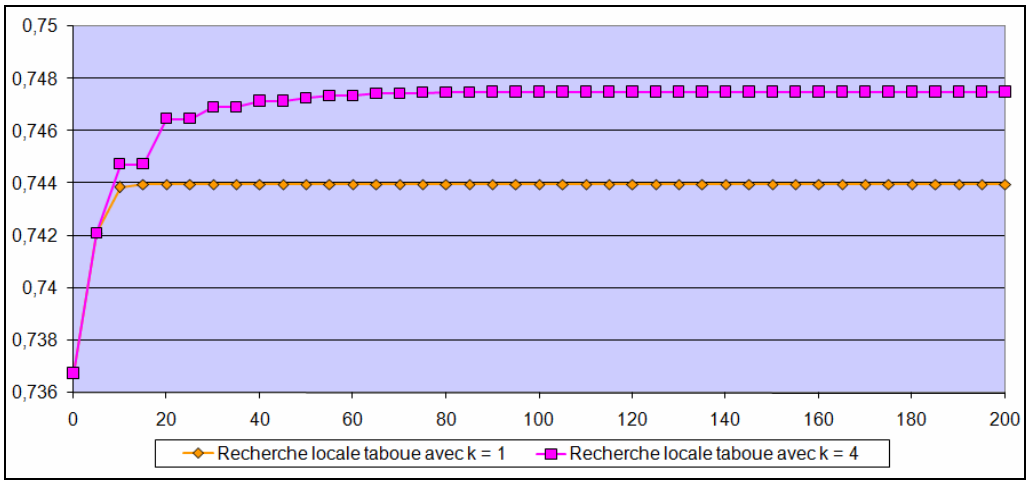


Figure F.36 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *défini par l'expert en cartographie* pour la recherche locale taboue (avec *k = 1* et *k = 4*)

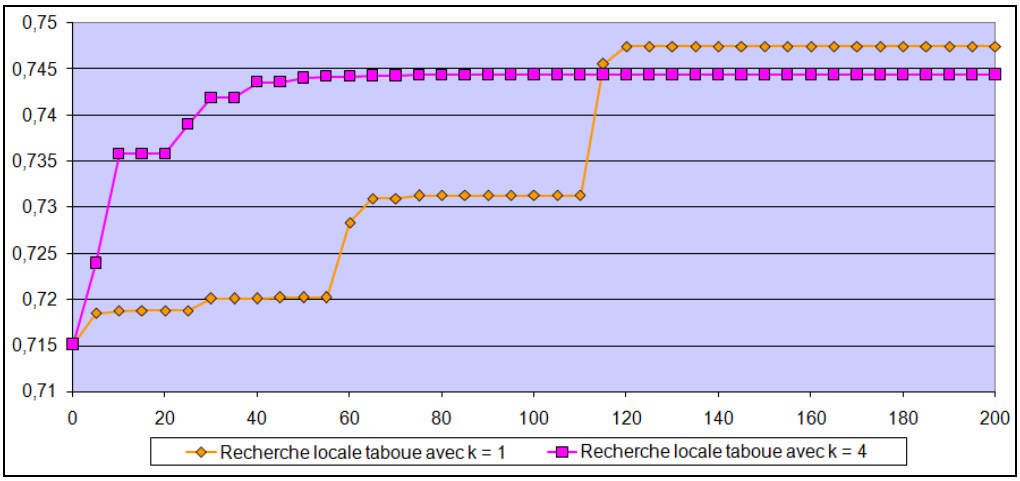


Figure F.37 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *défini par l'expert du modèle AGENT* pour la recherche locale taboue (avec *k = 1* et *k = 4*)

Les figures F.38, F.39 et F.40 présentent les résultats obtenus avec l'approche agent. Les meilleurs résultats obtenus pour cette approche l'ont été avec un coefficient de filtrage *CF* de 0.5. Ce coefficient a permis de trouver, pour les trois jeux de connaissances, une meilleure solution de qualité proche des qualités obtenues avec la recherche locale taboue avec k = 5. Le temps de convergence fut néanmoins plus long avec l'approche agent. Les résultats obtenus avec les deux autres valeurs de filtrage sont plutôt faibles. En effet, la valeur de filtrage CF = 0.25 n'a pas permis de filtrer efficacement le nombre de mouvements testés à chaque étape et le processus d'exploration a donc passé beaucoup de temps à chaque étape à définir le meilleur mouvement possible. Ces temps importants nécessaires aux tests du voisinage expliquent les longues périodes de stagnation visibles sur les courbes. Concernant la valeur de filtrage CF = 0.75, celle-ci a introduit un filtrage trop fort de l'ensemble des mouvements possibles et a entraîné la suppression de mouvements intéressants pour la recherche d'une meilleure solution.

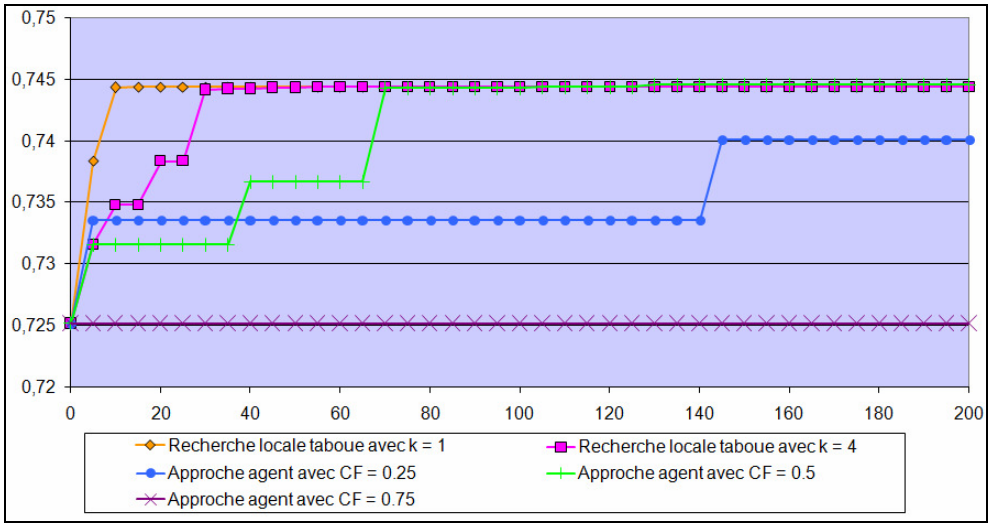


Figure F.38 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *basique* pour l'approche agent (avec *CF = 0.25*, *CF = 0.5* et *CF = 0.75*)

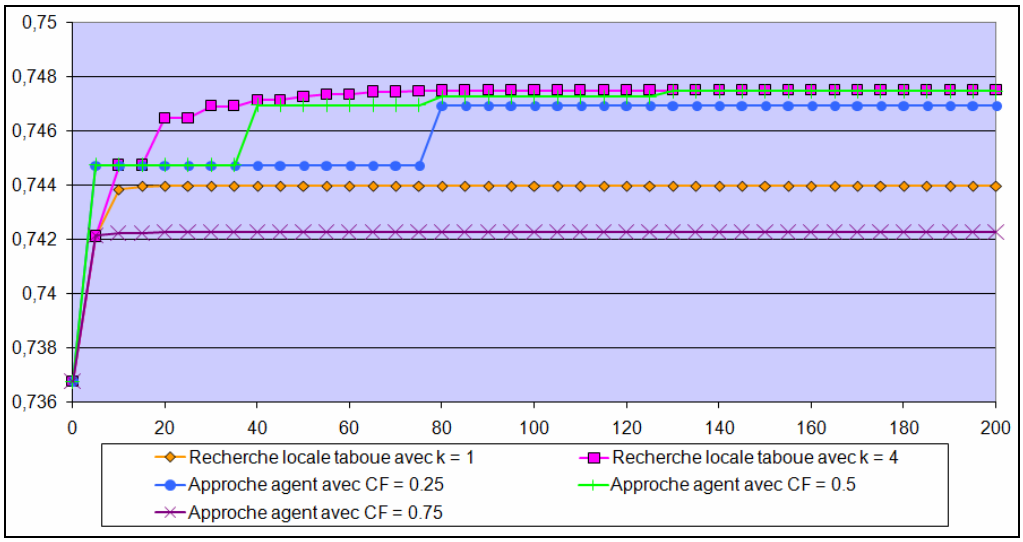


Figure F.39 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *défini par l'expert en cartographie* pour l'approche agent (avec *CF = 0.25*, *CF = 0.5* et *CF = 0.75*)

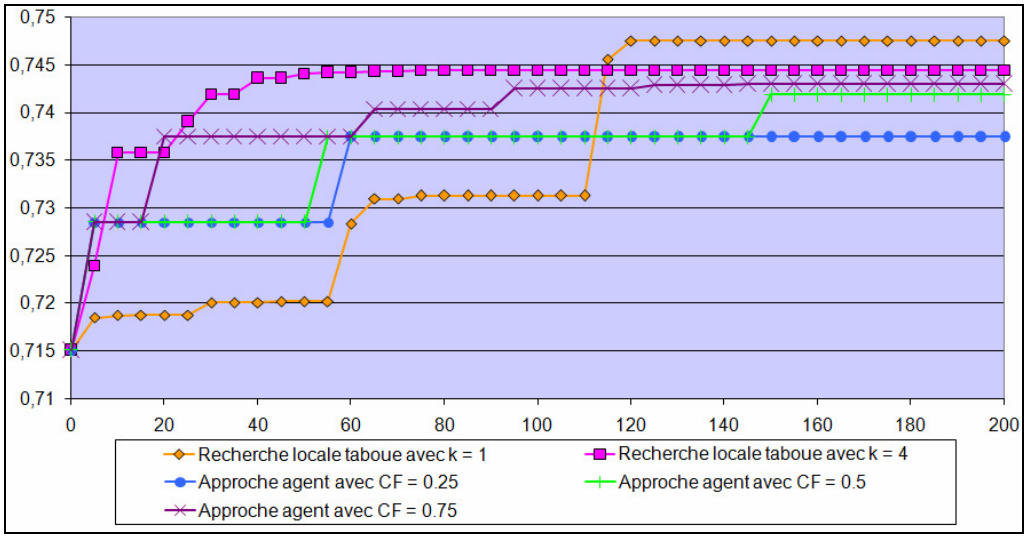


Figure F.40 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *défini par l'expert du modèle AGENT* pour l'approche agent (avec *CF = 0.25*, *CF = 0.5* et *CF = 0.75*)

Les figures F.41, F.42 et F.43 présentent les résultats obtenus avec l'approche combinant la recherche locale taboue avec un filtrage agent. Nous pouvons constater que le filtrage agent a bien permis de converger plus vite vers la meilleure solution. Si le filtrage avec une valeur *CF = 0.1* a bien permis de trouver des meilleures solutions totalement similaires en termes de qualité à l'approche sans filtrage, le filtrage avec une valeur de CF = 0.5 a entraîné pour deux des jeux de connaissances une légère détérioration de la qualité de la meilleure solution trouvée.

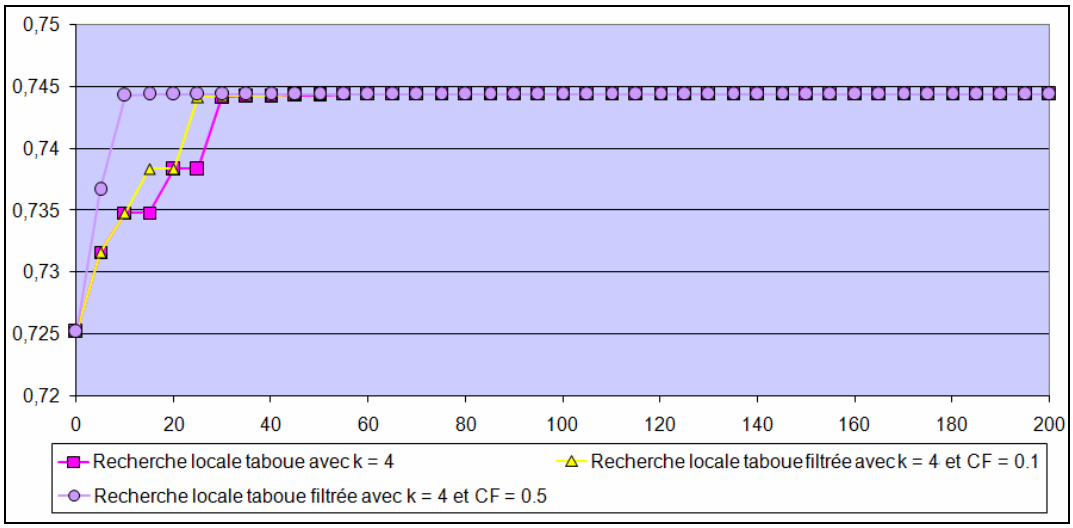


Figure F.41 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *basique* pour l'approche agent combinée à la recherche locale taboue (*CF = 0.1* et *CF = 0.5*)

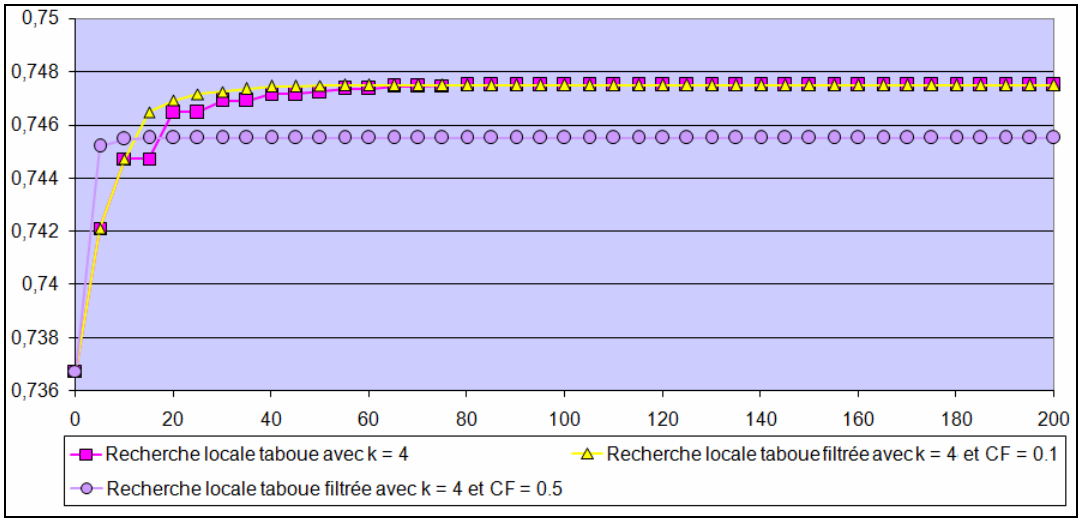


Figure F.42 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *défini par l'expert en cartographie* pour l'approche agent combinée à la recherche locale taboue (*CF = 0.1* et *CF = 0.5*)

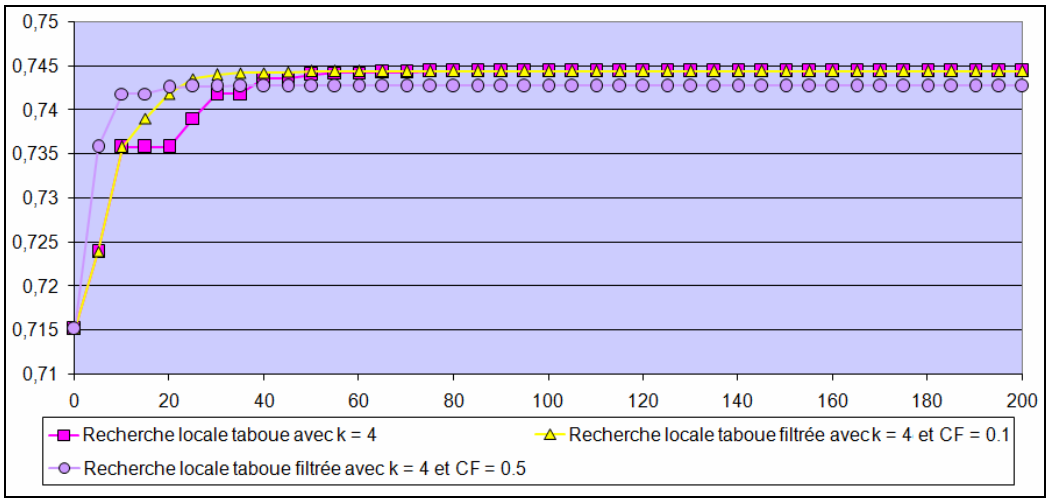


Figure F.43 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à la priorité des contraintes du jeu de connaissances *défini par l'expert du modèle AGENT* pour l'approche agent combinée à la recherche locale taboue (*CF = 0.1* et *CF = 0.5*)

En conclusion nous pouvons dire que l'approche agent s'est montrée moins performante dans ces expérimentations que l'approche locale taboue à cause de la taille importante de l'espace de recherche (en particulier le nombre de partitions en jeu). Concernant l'approche locale, cette expérimentation n'a pas permis de dégager de meilleure valeur pour la taille k de son voisinage. Enfin, le filtrage agent combiné à la recherche locale taboue a permis d'obtenir des résultats plutôt encourageants. Nous utilisons dans la suite de cette expérimentation sur la révision des connaissances relatives à la priorité des contraintes cette approche combinée en utilisant une taille de voisinage $k = 4$ et un coefficient de filtrage $CF = 0.1$.

Résultats de la révision des connaissances relatives à la priorité des contraintes

Les tables suivantes donnent les règles de priorité obtenues après révision (avec $CQ = 0$). Nous n'avons pas fait figurer dans ce tableau les trois contraintes de conservation. Celles-ci ne proposant pas d'action, leur priorité ne joue donc aucun rôle dans le processus de généralisation.

		Contrainte de satisfaction des bâtiments
Jeu de connaissances basique	Avant révision	*priorité* = 1
	Révision avec CQ = 0	**Si** $satisfaction_{satisfactionBati} < 4$ **alors** *priorité* = 2 **Si** $satisfaction_{satisfactionBati} \geq 4$ **et** $satisfaction_{satisfactionBati} < 5$ **alors** *priorité* = 1 **Si** $satisfaction_{satisfactionBati} \geq 5$ **et** $satisfaction_{satisfactionBati} < 6$ **alors** *priorité* = 2 **Si** $satisfaction_{satisfactionBati} \geq 6$ **et** $satisfaction_{satisfactionBati} < 7$ **alors** *priorité* = 1 **Si** $satisfaction_{satisfactionBati} \geq 7$ **alors** *priorité* = 2
Jeu de connaissances défini par l'expert en cartographie	Avant révision	**Si** $satisfaction_{satisfactionBati} < 5$ **alors** *priorité* = 4 **Si** $satisfaction_{satisfactionBati} \geq 5$ **et** $satisfaction_{satisfactionBati} < 8$ **alors** *priorité* = 3 **Si** $satisfaction_{satisfactionBati} \geq 8$ **alors** *priorité* = 2

	Révision avec CQ = 0	**Si** *satisfaction*$_{satisfactionBati}$ < 5 **alors** *priorité* = 4 **Si** *satisfaction*$_{satisfactionBati}$ ≥ 5 **et** satisfaction$_{satisfactionBati}$ < 8 **alors** *priorité* = 3 **Si** *satisfaction*$_{satisfactionBati}$ ≥ 8 **alors** *priorité* = 2
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	**Si** *satisfaction*$_{satisfactionBati}$ < 8 **alors** *priorité* = 3 **Si** *satisfaction*$_{satisfactionBati}$ ≥ 8 **alors** *priorité* = 1
	Révision avec CQ = 0	**Si** *satisfaction*$_{satisfactionBati}$ < 5 **alors** *priorité* = 3 **Si** *satisfaction*$_{satisfactionBati}$ ≥ 5 **et** satisfaction$_{satisfactionBati}$ < 6 **alors** *priorité* = 5 **Si** *satisfaction*$_{satisfactionBati}$ ≥ 6 **alors** *priorité* = 4

		Contrainte de proximité
Jeu de connaissances basique	Avant révision	*priorité* = 1
	Révision avec CQ = 0	**Si** *satisfaction*$_{proximité}$ < 2 **alors** *priorité* = 2 **Si** *satisfaction*$_{proximité}$ ≥ 2 **et** *satisfaction*$_{proximité}$ < 3 **alors** *priorité* = 1 **Si** *satisfaction*$_{proximité}$ ≥ 3 **et** *satisfaction*$_{proximité}$ < 6 **alors** *priorité* = 2 **Si** *satisfaction*$_{proximité}$ ≥ 6 **alors** *priorité* = 1
Jeu de connaissances défini par l'expert en cartographie	Avant révision	**Si** *satisfaction*$_{proximité}$ < 8 **alors** *priorité* = 5 **Si** *satisfaction*$_{proximité}$ ≥ 8 **alors** *priorité* = 4
	Révision avec CQ = 0	*priorité* = 2
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	**Si** *satisfaction*$_{proximité}$ < 8 **alors** *priorité* = 4 **Si** *satisfaction*$_{proximité}$ ≥ 8 **alors** *priorité* = 2
	Révision avec CQ = 0	**Si** *satisfaction*$_{proximité}$ < 6 **alors** *priorité* = 4 **Si** *satisfaction*$_{proximité}$ ≥ 6 **alors** *priorité* = 2

		Contrainte de densité
Jeu de connaissances basique	Avant révision	*priorité* = 1
	Révision avec CQ = 0	*priorité* = 1
Jeu de connaissances défini par l'expert en	Avant révision	**Si** *satisfaction*$_{densité}$ < 4 **alors** *priorité* = 3 **Si** *satisfaction*$_{densité}$ ≥ 4 **alors** *priorité* = 1

cartographie	Révision avec CQ = 0	**Si** $satisfaction_{densité} < 2$ **alors** *priorité* = 1 **Si** $satisfaction_{densité} \geq 2$ **et** $satisfaction_{densité} < 4$ **alors** *priorité* = 3 **Si** $satisfaction_{densité} \geq 4$ **alors** *priorité* = 1
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	*priorité* = 5
	Révision avec CQ = 0	**Si** $satisfaction_{densité} < 6$ **alors** *priorité* = 5 **Si** $satisfaction_{densité} \geq 6$ **alors** *priorité* = 1

Nous remarquons sur ces règles de priorité que, malgré le coefficient de qualité des connaissances égal à 0, les règles obtenues après révision conservent des similarités importantes avec les règles initiales. Dans l'ensemble, les règles obtenues après révision tendent à pousser l'agent *groupement de bâtiments* à traiter en priorité, lorsque les trois contraintes sont très insatisfaites, les problèmes liés à la contrainte de satisfaction des bâtiments (sauf pour les règles obtenues avec le jeu de connaissances *défini par l'expert du modèle AGENT* qui tend plutôt à privilégier la contrainte de densité). Les problèmes liés à la contrainte de densité sont généralement traités en dernier.

Les figures F.45, F.46 et F47 présentent les résultats obtenus avant et après révision sur la zone de test. Ces résultats montrent que la révision de priorité a bien permis au système de généralisation de converger plus rapidement vers le meilleur état.

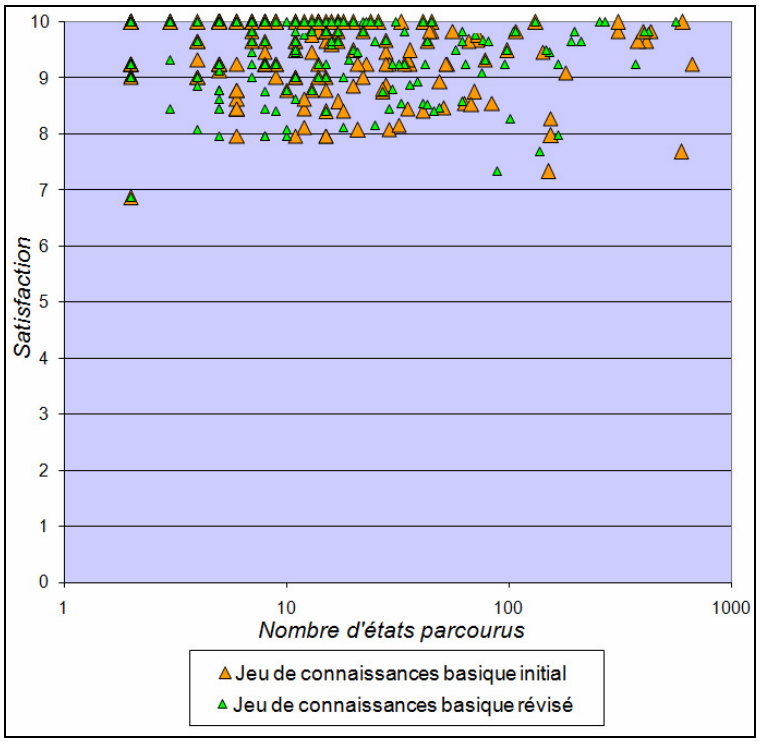


Figure F.44 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) des connaissances relatives à la priorité des contraintes (échelle logarithmique pour le nombre d'états parcourus)

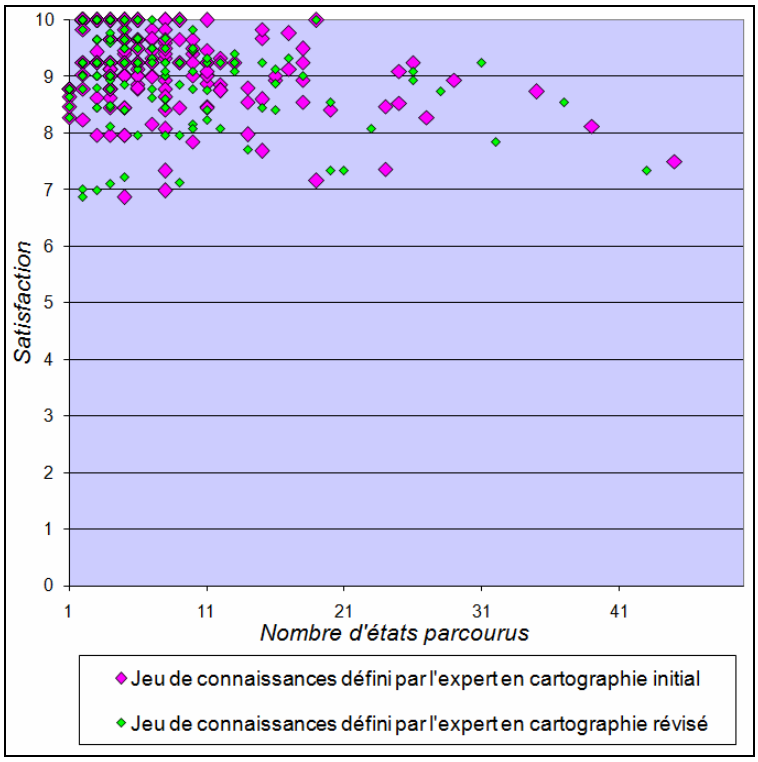


Figure F.45 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) des connaissances relatives à la priorité des contraintes

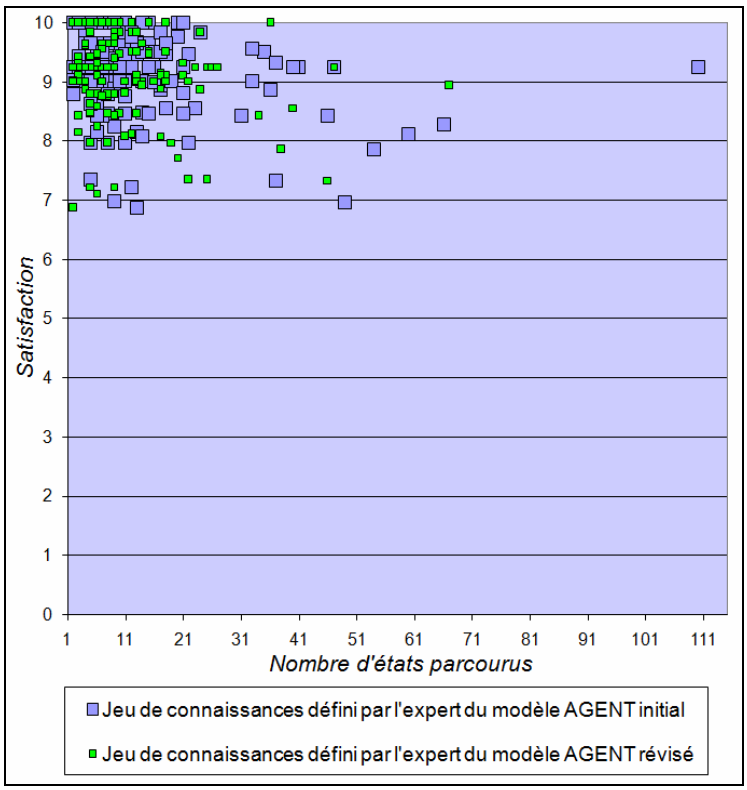


Figure F.46 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (avec CQ = 0) des connaissances relatives à la priorité des contraintes

Les tables suivantes donnent les résultats des différents jeux de connaissances obtenus après révision sur la zone de test.

Révision de la connaissance relative à la priorité des contraintes pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	0.862	0.484	0.787
Après révision avec ***CQ*** *= 0*	0.862	0.510	0.792
Après révision avec ***CQ*** *= 0.01*	0.862	0.506	0.791
Après révision avec ***CQ*** *= 0.1*	0.862	0.484	0.787

Révision de la connaissance relative à la priorité des contraintes pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	0.807	0.753	0.796
Après révision avec ***CQ*** *= 0*	0.81	0.796	0.807
Après révision avec ***CQ*** *= 0.01*	0.817	0.789	0.812
Après révision avec ***CQ*** *= 0.1*	0.816	0.769	0.806

Révision de la connaissance relative à la priorité des contraintes pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	0.838	0.681	0.807
Après révision avec ***CQ*** *= 0*	0.838	0.724	0.815
Après révision avec ***CQ*** *= 0.01*	0.837	0.715	0.813
Après révision avec ***CQ*** *= 0.1*	0.839	0.686	0.808

Ces tables confirment bien la tendance observée sur les nuages de points. La révision des connaissances relatives à la priorité des contraintes a bien permis au système de généralisation de converger plus vite vers le meilleur état.

F.III.6.3 Bilan de l'expérimentation

Cette expérimentation a permis de valider l'application de notre approche de révision pour les connaissances relatives à la priorité des contraintes. Notre approche a en effet permis d'améliorer les trois jeux de connaissances.

Cette expérimentation a également permis de tester nos diverses approches d'exploration. Parmi celles-ci, la plus prometteuse semble être notre approche mêlant une métaheuristique classique de recherche locale à un filtrage agent du voisinage qui a permis de trouver de bonnes solutions tout en convergeant plus rapidement.

F.III.7 Révision des connaissances relatives à l'application des actions

F.III.7.1 Contexte de l'expérimentation

Pour chaque action, une base de règles permet de déterminer si, pour un état donné de l'agent géographique (caractérisé par l'ensemble des mesures liées à la contrainte qui propose l'action), l'action doit être appliquée ou non et avec quel poids (cf. B.V.3.1.1). La connaissance relative à l'application des actions est représentée sous la forme d'une base de règles de $\mathcal{BR}$ (cf. D.IV.3.2).

Nous proposons d'utiliser pour ce type de connaissances la même approche que pour les autres connaissances représentées sous forme de base de règles de $\mathcal{BR}$. Nous utilisons pour ces expérimentations l'approche de partitionnement par apprentissage et comparaison de règles. En effet, cette approche a donné de bons résultats durant les expérimentations précédentes et offre l'avantage de générer peu de partitions, ce qui est particulièrement important, vu que la combinatoire est très importante. Nous disposons en effet de 5 actions. Or, chaque partition peut prendre 6 valeurs de poids (entier compris entre 0 et 5). La taille de l'espace de recherche est donc de :

$$\prod_{act \in Ensemble\_des\_actions} 6^{nbPartition(act)}$$

Nous allons, tout comme pour les connaissances relatives à la priorité des contraintes, nous intéresser particulièrement dans le cadre de cette expérimentation à l'étape d'exploration de l'espace des affectations possibles de conclusion. Nous testons ainsi, au cours de cette expérimentation, les diverses approches que nous avons proposées. La taille maximale du voisinage est ici de 5. En effet, le poids des actions varie entre 0 (l'action n'est jamais proposée) et 5. Nous testerons donc l'approche de recherche locale taboue avec une valeur de *k* égale 1 et égale à 6.

Cette connaissance a à la fois un rôle d'élagage et un rôle de guidage. Sa révision ne permettra pas de gain d'efficacité pour le jeu de connaissances *basique* pour lequel toutes les actions sont proposées pour tous les états et qui utilise le critère d'élagage ***CEPAC***. Elle a peu de chance d'améliorer l'efficacité obtenue avec le jeu de connaissances *défini par l'expert du modèle AGENT* pour lequel toutes les actions sont proposées pour tous les états à l'exception de l'action de suppression globale de bâtiments qui n'est pas proposée lorsque le *groupement de bâtiments* est composé d'un seul bâtiment. Or, dans ce cas, la contrainte de densité a presque toujours une satisfaction égale à 10 et ne propose donc de toute façon pas d'action. Le seul jeu de connaissances pour lequel la révision de cette connaissance a une bonne chance d'améliorer l'efficacité est le jeu de connaissances *défini par l'expert en cartographie* qui comporte des règles assez restrictives sur l'application des actions.

F.III.7.2 Résultats obtenus

Comparaison des approches d'exploration

Nous nous intéressons dans un premier temps aux résultats obtenus lors de l'exploration de l'espace des affectations possibles de conclusion. Toutes les expérimentations menées dans ce cadre l'ont été avec un *coefficient de qualité des connaissances* nul.

Les figures F.47, F.48 et F.49 présentent les résultats obtenus avec les deux valeurs de *k* testées. Nous remarquons sur ces figures que les résultats obtenus avec la valeur de *k = 5* sont meilleurs et cela pour les trois jeux de connaissances. En effet, la meilleure solution trouvée avec cette valeur de *k* est supérieure en termes de valeur de performance. Comme pour le cas du jeu de connaissances *défini par l'expert du modèle AGENT* pour les connaissances relatives à la priorité des contraintes, l'obtention de la meilleure solution a nécessité de modifier de façon importante la valeur de conclusion de seulement quelques partitions et donc de permettre à la recherche menée avec la valeur de *k* égale à 5 de converger plus rapidement vers la meilleure solution.

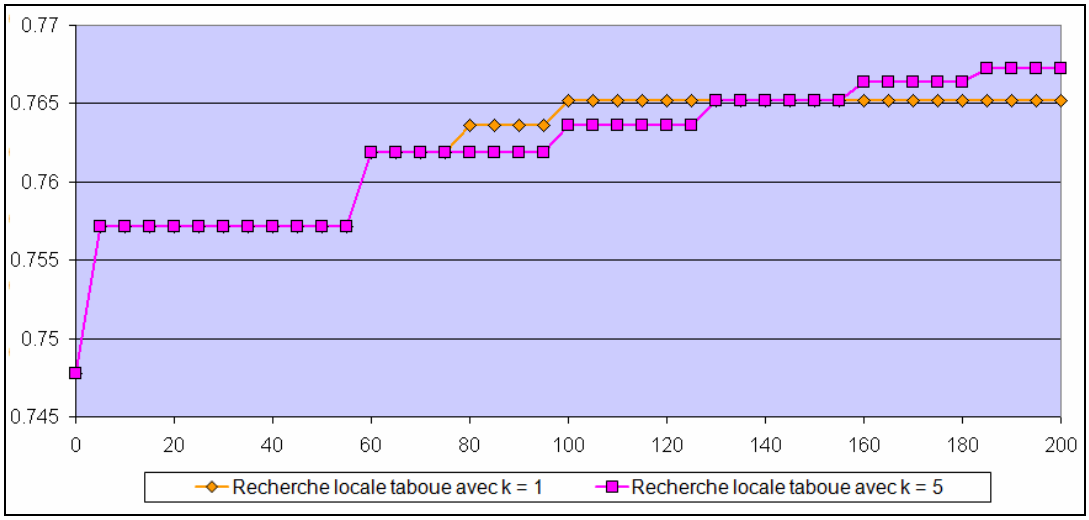


Figure F.47 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *basique* pour la recherche locale taboue (avec *k = 1* et *k = 5*)

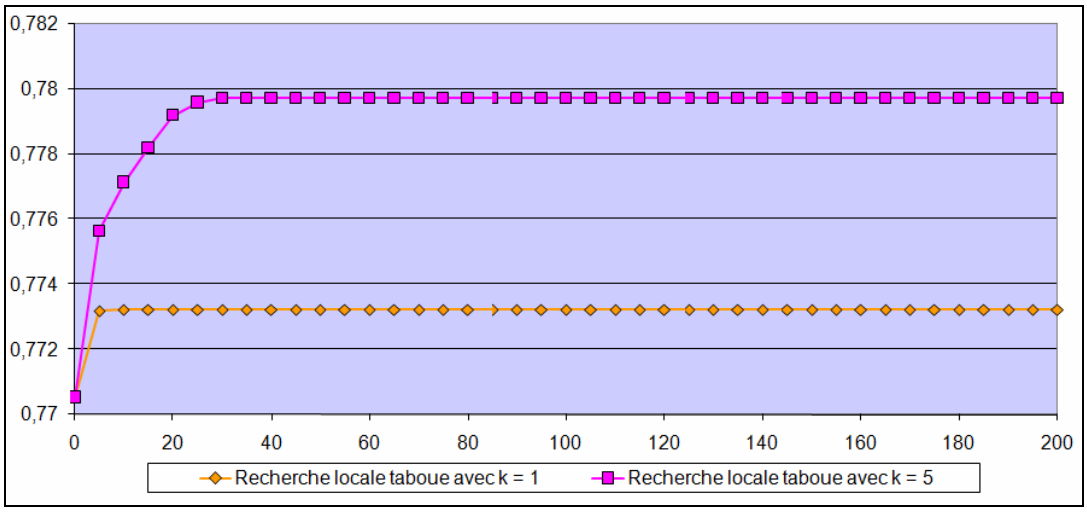


Figure F.48 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *défini par l'expert en cartographie* pour la recherche locale taboue (avec *k = 1* et *k = 5*)

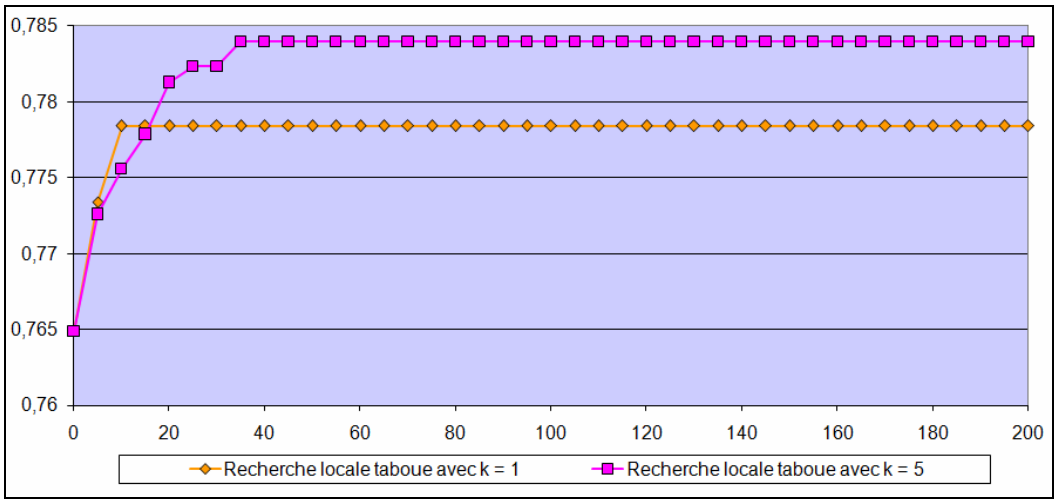


Figure F.49 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *défini par l'expert du modèle AGENT* pour la recherche locale taboue (avec *k = 1* et *k = 5*)

Les figures F.50, F.51 et F.52 présentent les résultats obtenus avec l'approche agent. Les meilleurs résultats obtenus pour cette approche l'ont été avec un coefficient de filtrage *CF = 0.25* qui a permis de trouver, pour les trois jeux de connaissances, de meilleures solutions que celles obtenues avec la recherche locale taboue avec *k = 5*. Le temps de convergence fut, avec cette approche agent, plus court pour le jeu de connaissances *basique* et celui *défini par l'expert en cartographie* mais plus long pour le troisième jeu de connaissances. L'utilisation d'une valeur de filtrage pour l'approche agent plus élevée a permis au processus d'exploration de converger plus rapidement vers la meilleure solution mais a entraîné des détériorations de qualité de la meilleure solution trouvée.

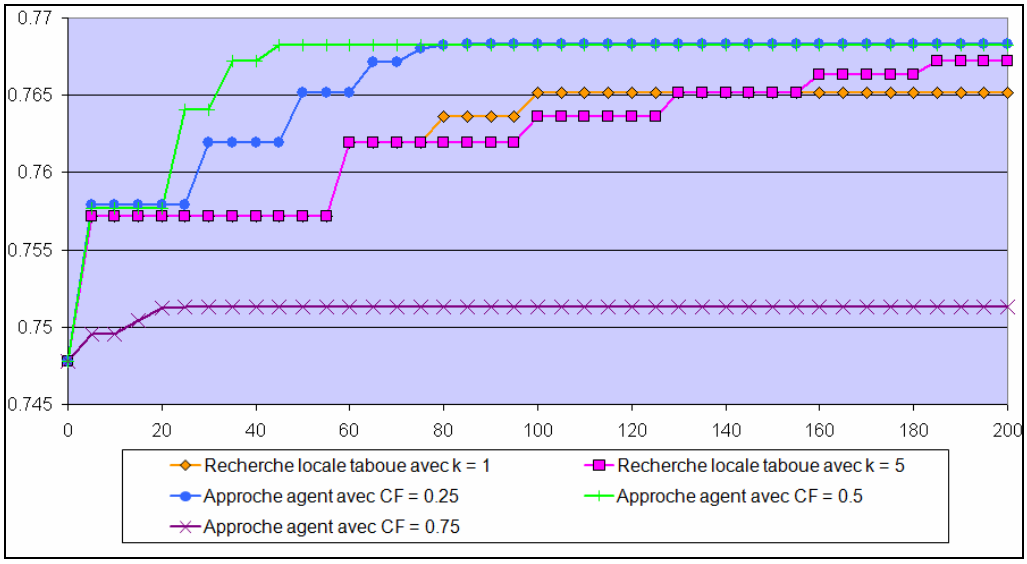


Figure F.50 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *basique* pour l'approche agent (avec *CF = 0.25*, *CF = 0.5* et *CF = 0.75*)

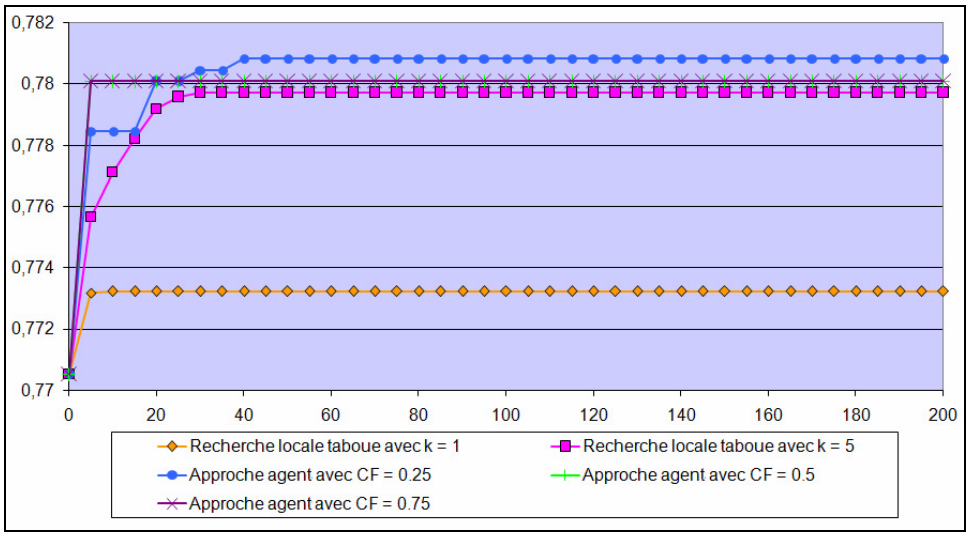


Figure F.51 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *défini par l'expert en cartographie* pour l'approche agent (avec *CF = 0.25*, *CF = 0.5* et *CF = 0.75*)

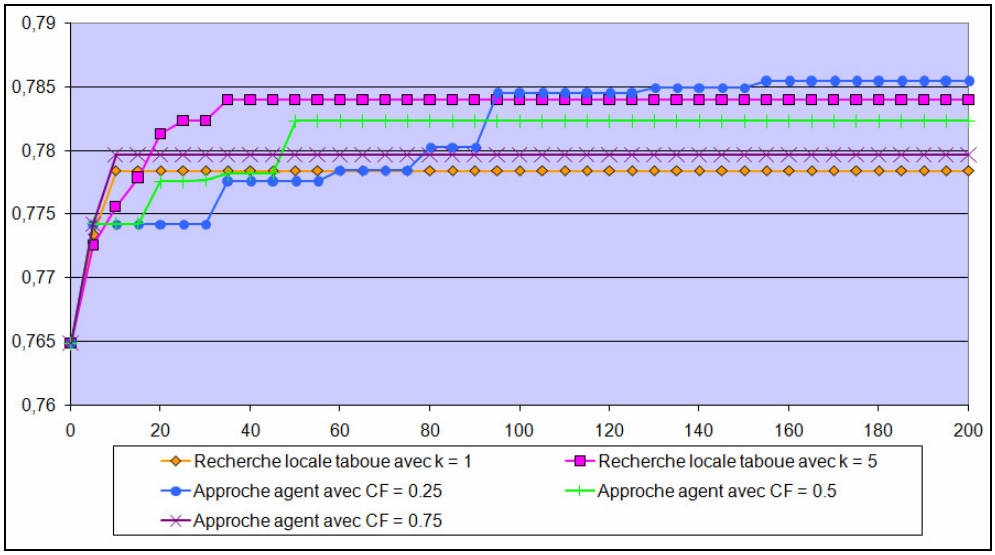


Figure F.52 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *défini par l'expert du modèle AGENT* pour l'approche agent (avec *CF = 0.25*, *CF = 0.5* et *CF = 0.75*)

Les figures F.53, F.54 et F.55 présentent les résultats obtenus avec l'approche combinant la recherche locale taboue à un filtrage agent. Nous constatons que le filtrage agent a bien permis, à l'image des résultats obtenus pour la révision des connaissances relatives à la priorité des contraintes, de converger plus rapidement vers la meilleure solution. Si le filtrage avec une valeur *CF = 0.1* a bien permis de trouver des meilleures solutions totalement similaires en termes de qualité à l'approche sans filtrage, le filtrage avec une valeur de *CF = 0.5* a entraîné pour les deux des trois jeux de connaissances une détérioration importante de la qualité de la meilleure solution trouvée. En effet, le filtrage avec une valeur de *CF = 0.5* a entraîné la suppression de certains mouvements de la liste des mouvements possibles a priori important pour trouver la meilleure solution.

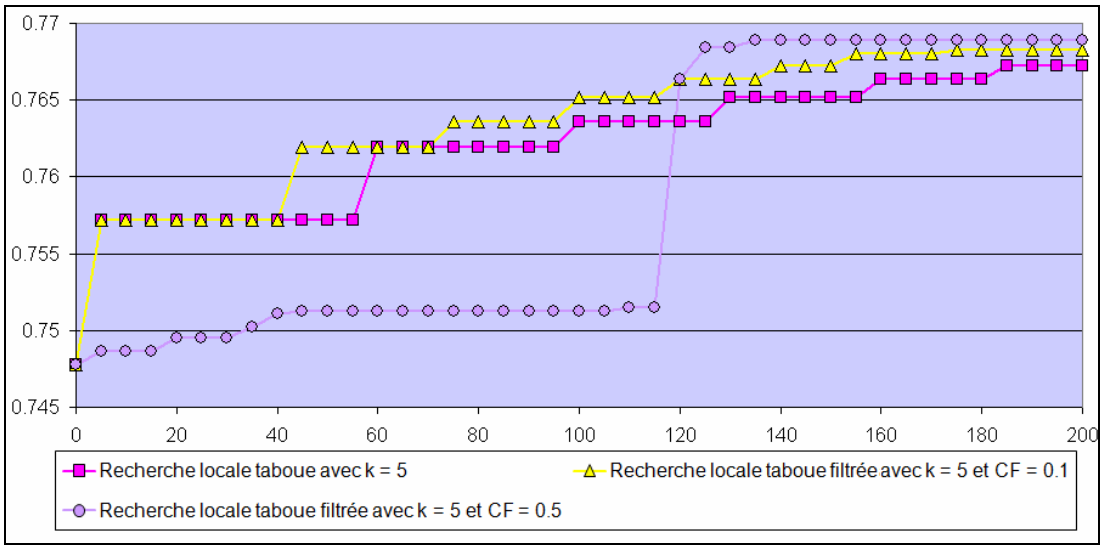


Figure F.53 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *basique* pour l'approche agent combinée à la recherche locale taboue (*CF = 0.1* et *CF = 0.5*)

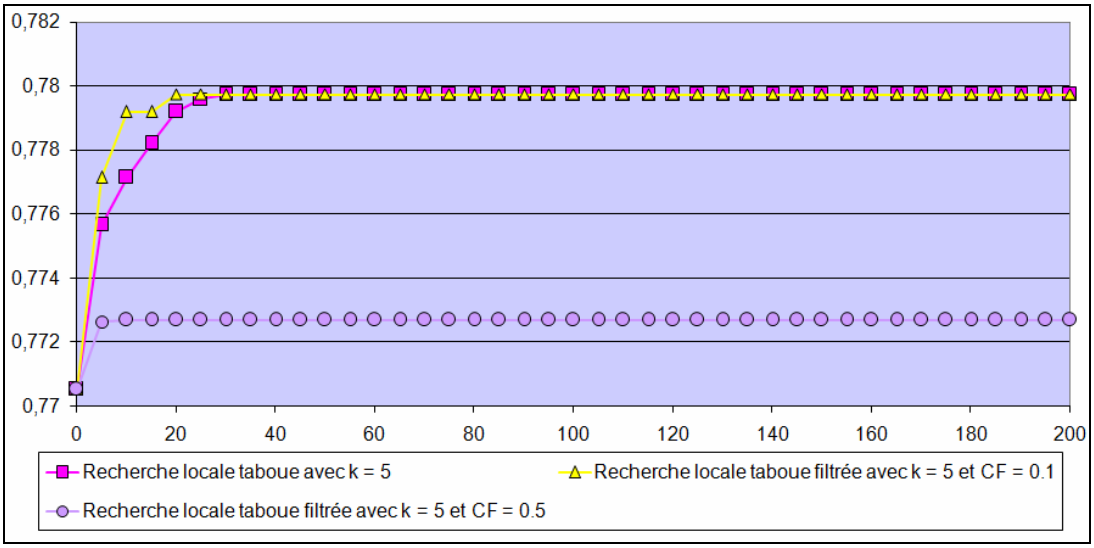


Figure F.54 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *défini par l'expert en cartographie* pour l'approche agent combinée à la recherche locale taboue (*CF = 0.1* et *CF = 0.5*)

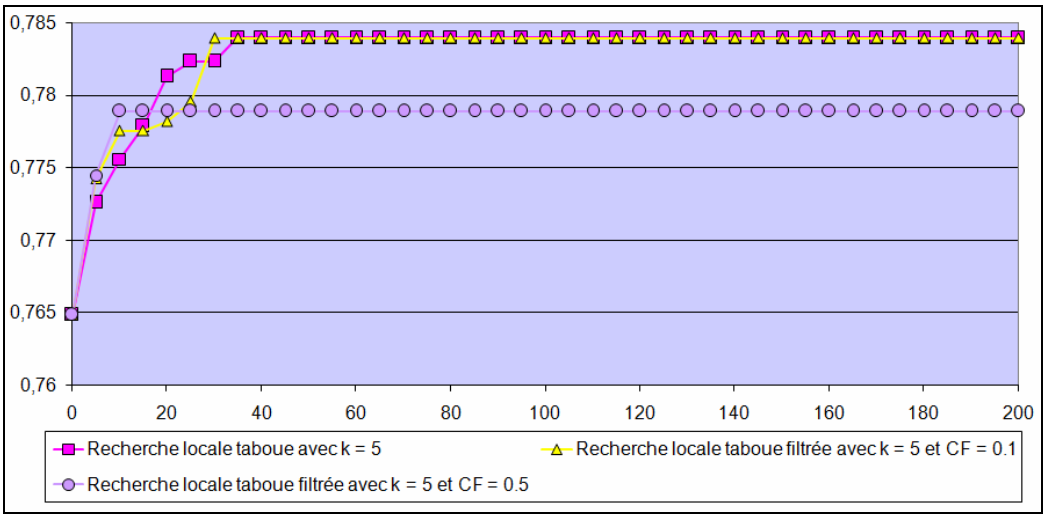


Figure F.55 Evolution des performances (en ordonnée) de la meilleure solution trouvée par rapport au temps (en abscisse) pour la révision des connaissances relatives à l'application des actions du jeu de connaissances *défini par l'expert du modèle AGENT* pour l'approche agent combinée à la recherche locale taboue (*CF = 0.1* et *CF = 0.5*)

En conclusion, l'approche agent s'est montrée plus performante pour ces expérimentations que l'approche locale taboue. Concernant cette dernière, la prise en compte d'un voisinage de taille plus importante a permis de largement améliorer les résultats obtenus avec celle-ci. Enfin, l'ajout d'un filtrage agent à l'approche locale taboue a permis à cette approche de converger plus rapidement vers la meilleure solution. Nous utilisons dans la suite de cette expérimentation sur la révision des connaissances relatives à l'application des actions l'approche combinée en utilisant une taille de voisinage $k = 5$ et un coefficient de filtrage $CF = 0.1$.

Résultats de la révision des connaissances relatives à l'application des actions

Les tables suivantes donnent les règles d'application des actions obtenues après révision (avec $CF = 0$).

		Action de généralisation des bâtiments
Jeu de connaissances basique	Avant révision	*poids* = 1
	Révision avec CQ = 0	*poids* = 1
Jeu de connaissances défini par l'expert en cartographie	Avant révision	*poids* = 1
	Révision avec CQ = 0	*poids* = 1
Jeu de	Avant révision	*poids* = 1

connaissances défini par l'expert du modèle AGENT	Révision avec CQ = 0	*poids* = 1

		Action de suppression globale de bâtiments
Jeu de connaissances basique	Avant révision	*poids* = 1
	Révision avec CQ = 0	**Si** *nb_bâtiments* > 14 **et** *densité* > 0.4 **alors** *poids = 1*
Jeu de connaissances défini par l'expert en cartographie	Avant révision	*poids* = 1
	Révision avec CQ = 0	*poids* = 1
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	**Si** *nb_bâtiments* > 1 **alors** *poids = 1*
	Révision avec CQ = 0	*Ne jamais proposer l'action*

		Action de déplacement de bâtiments
Jeu de connaissances basique	Avant révision	*poids* = 1
	Révision avec CQ = 0	**Si** *minProximité* > *-23.5* **alors** *poids* = 1
Jeu de connaissances défini par l'expert en cartographie	Avant révision	*poids* = 3
	Révision avec CQ = 0	**Si** *prem_quartileProximité* > *-6.18* **et** $satisfaction_{proximité}$ < 2 **alors** *poids* = 3 **Si** *prem_quartileProximité* ≤ *-6.18* **et** $satisfaction_{proximité}$ < 6 **alors** *poids* = 3 **Si** *prem_quartileProximité* > *-6.18* **et** $satisfaction_{proximité}$ < 3 **et** *medianeProximité* > *1.9* **alors** *poids* = 3
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	**Si** $satisfaction_{proximité}$ < 6 **alors** *poids* = 2 **Si** $satisfaction_{proximité}$ ≥ 6 **alors** *poids* = 4
	Révision avec CQ = 0	**Si** $satisfaction_{proximité}$ < 3 **alors** *poids* = 2 **Si** $satisfaction_{proximité}$ ≥ 4 **et** $satisfaction_{proximité}$ < 7 **alors** *poids* = 4

		Action de suppression locale de bâtiments
Jeu de connaissances basique	Avant révision	*poids* = 1
	Révision avec CQ = 0	*poids* = 1
Jeu de connaissances défini par l'expert en cartographie	Avant révision	**Si** $satisfaction_{proximité} < 6$ **alors** *poids* = 1
	Révision avec CQ = 0	**Si** $satisfaction_{proximité} < 6$ **alors** *poids* = 1
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	**Si** $satisfaction_{proximité} < 6$ **alors** *poids* = 3 **Si** $satisfaction_{proximité} \geq 6$ **alors** *poids* = 2
	Révision avec CQ = 0	**Si** $satisfaction_{proximité} < 6$ **alors** *poids* = 3

		Action de suppression recentrage d'un bâtiment
Jeu de connaissances basique	Avant révision	*poids* = 1
	Révision avec CQ = 0	*Ne jamais proposer l'action*
Jeu de connaissances défini par l'expert en cartographie	Avant révision	**Si** $satisfaction_{proximité} < 6$ **alors** *poids* = 2
	Révision avec CQ = 0	*Ne jamais proposer l'action*
Jeu de connaissances défini par l'expert du modèle AGENT	Avant révision	**Si** $satisfaction_{proximité} < 6$ **alors** *poids* = 4 **Si** $satisfaction_{proximité} \geq 6$ **alors** *poids* = 3
	Révision avec CQ = 0	**Si** $satisfaction_{proximité} \geq 6$ **alors** *poids* = 3

Nous pouvons constater que la révision des connaissances relatives à l'application des actions a permis de diminuer le nombre d'actions proposées à chaque état. Si les actions de déplacement de bâtiments et de généralisation des bâtiments sont elles proposées pour (presque) tous les états, ce n'est pas le cas des autres actions.

Enfin concernant l'ordonnancement des actions proposées par la contrainte de proximité, les bases de règles révisées proposent d'appliquer en priorité l'action de déplacement.

Les figures F.56, F.57 et F.58 présentent les résultats obtenus avant et après révision sur la zone de test. Ces résultats montrent que la révision de priorité a bien permis un gain en efficience pour le système de généralisation (beaucoup plus de points situés sur la gauche).

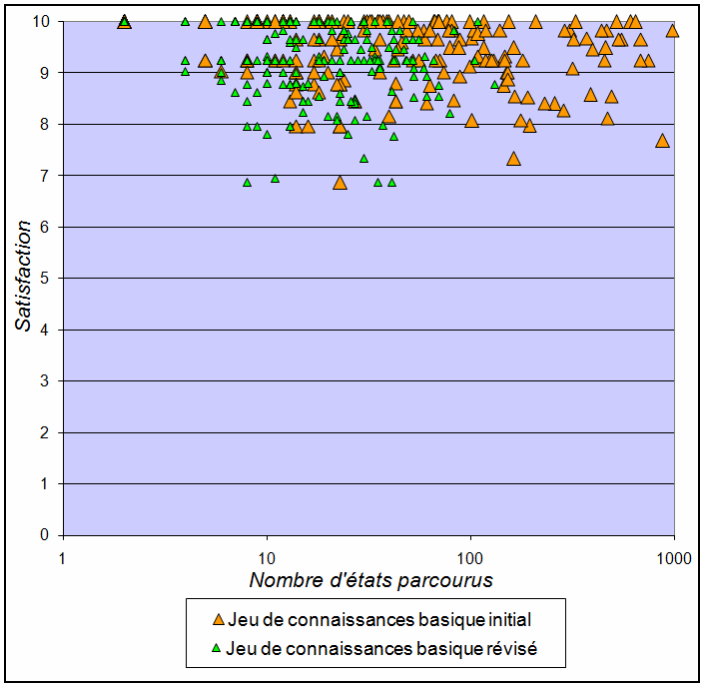


Figure F.56 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) des connaissances relatives à l'application des actions (échelle logarithmique pour le nombre d'états parcourus)

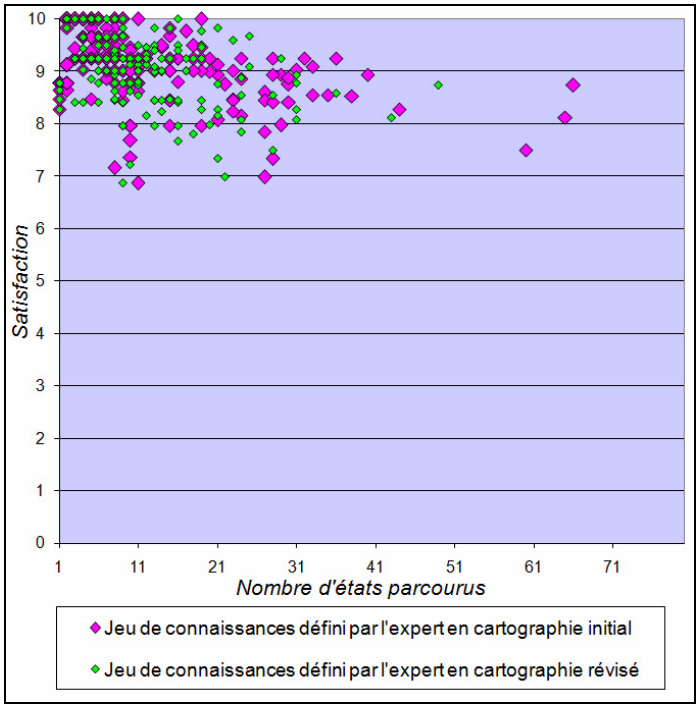


Figure F.57 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) des connaissances relatives à l'application des actions

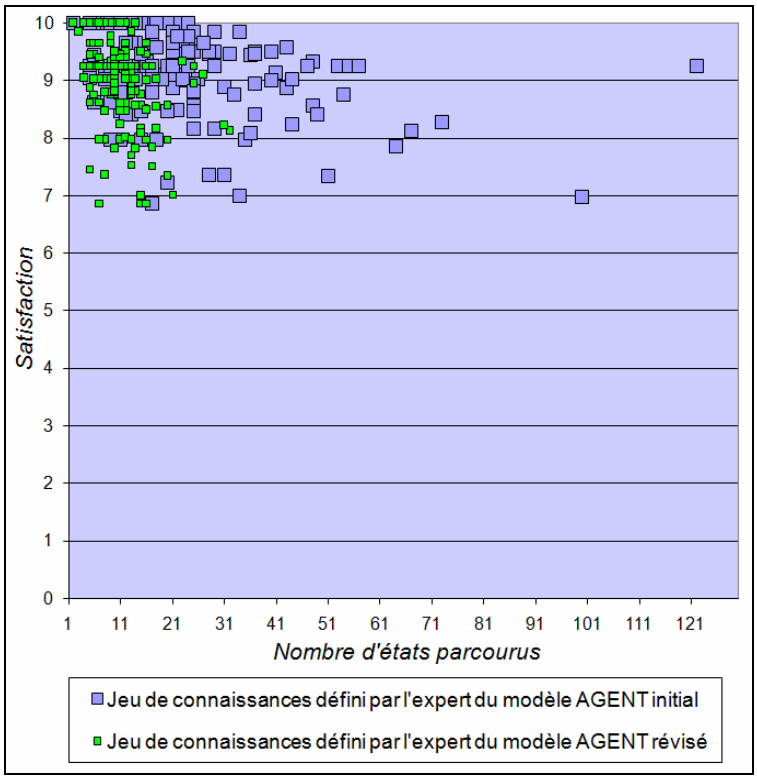


Figure F.57 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (avec CQ = 0) des connaissances relatives à l'application des actions

Concernant les résultats cartographiques obtenus sur la zone de test avant et après révision (avec un coefficient de qualité de la connaissance de 0) avec le jeu de connaissances *basique* (figure F.58) nous observons une légère baisse de qualité. En effet, quelques problèmes de proximité et de densité sont apparus. Au contraire, pour le jeux de connaissances *défini par l'expert en cartographie* (figure F.59), la révision des connaissances relatives à l'application des actions a permis d'améliorer sensiblement le résultat visuel en faisant disparaître la plupart des problèmes de proximité et de densité et même quelques problèmes de bâtiments non généralisés. Enfin, pour le jeu de connaissances *défini par l'expert du modèle AGENT* (figure F.60), les résultats obtenus avec les connaissances révisées sont proches des résultats obtenus avec les connaissances initiales (très légèrement inférieurs).

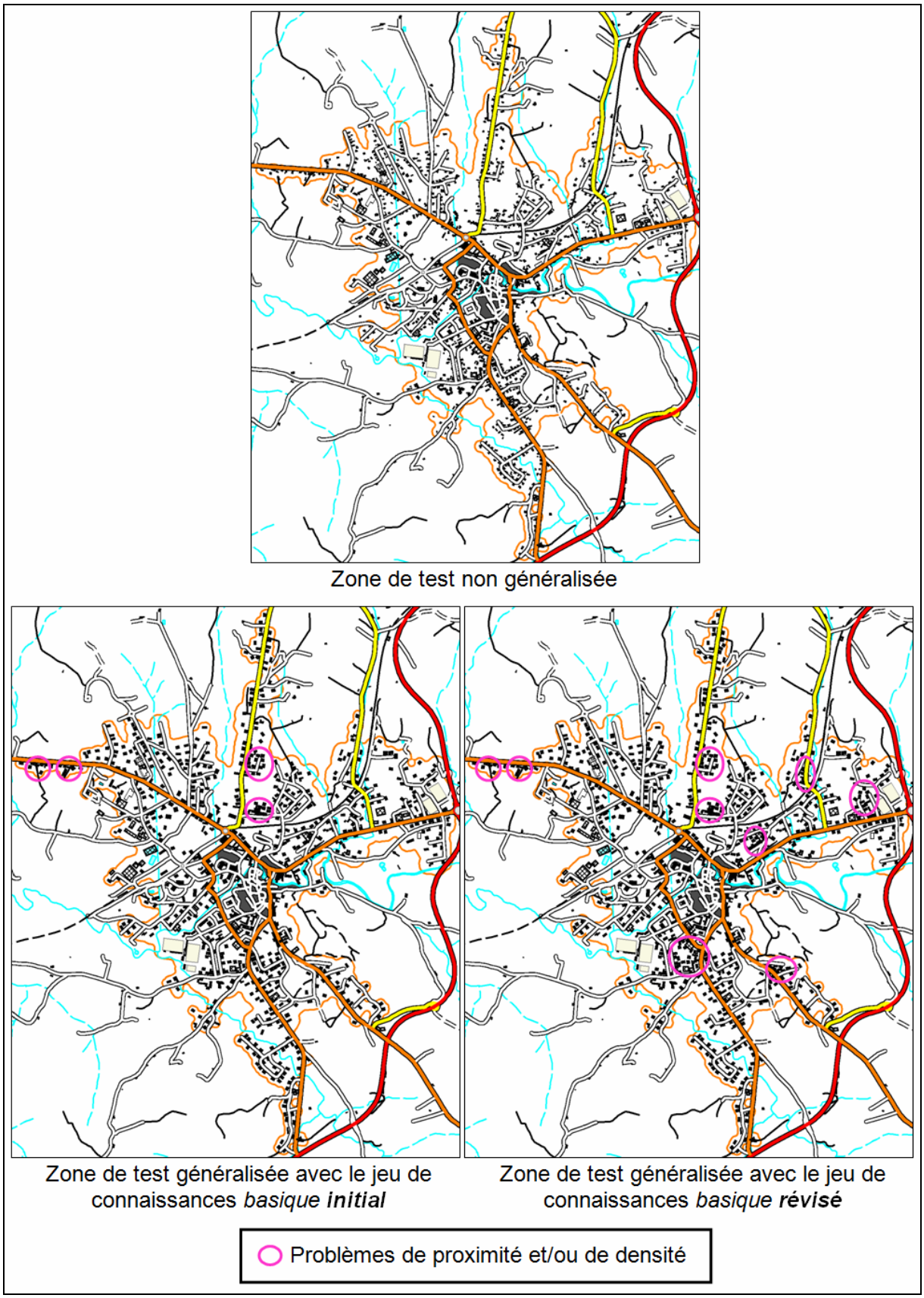


Figure F.58 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision (avec CQ = 0) des connaissances relatives à l'application des actions

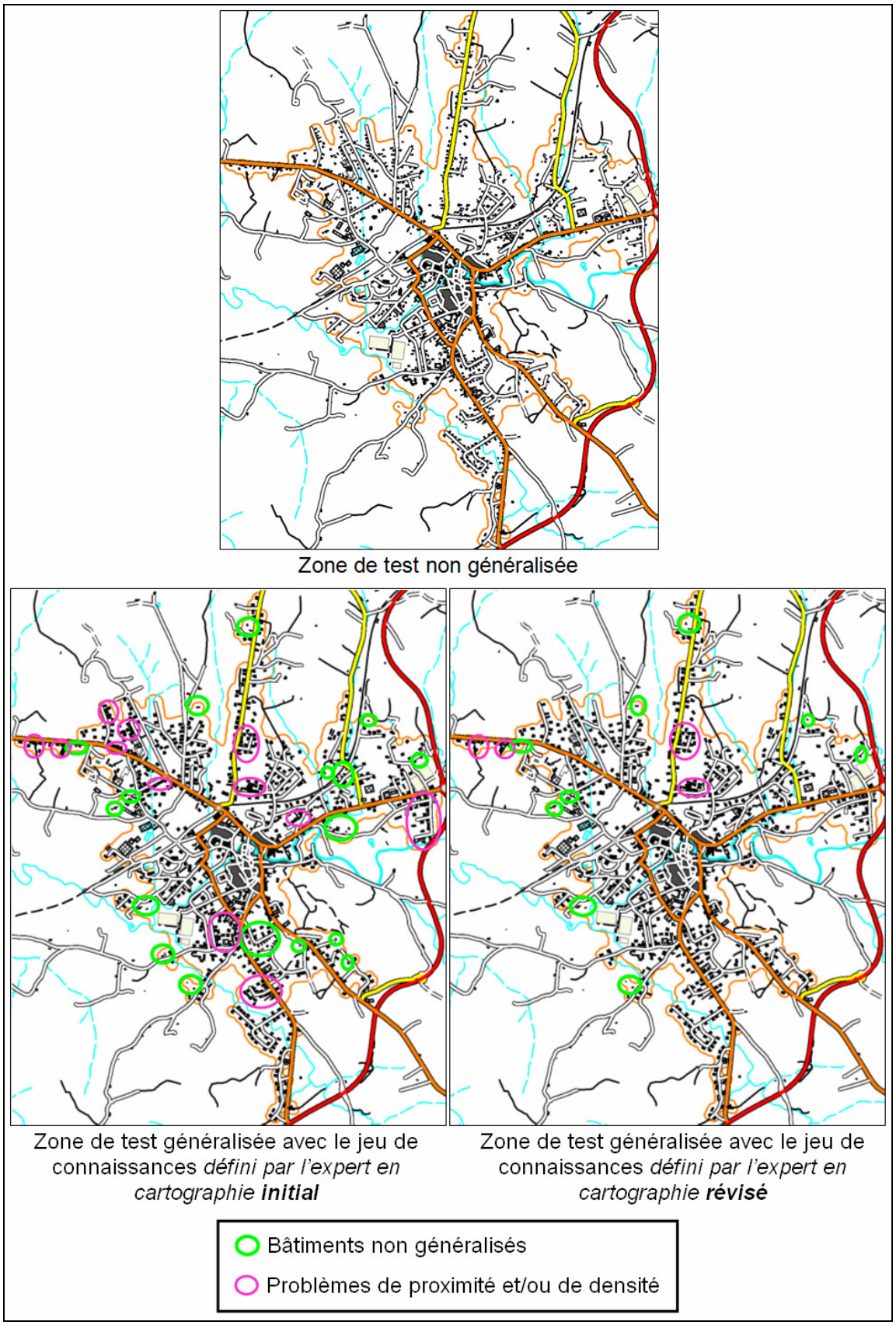


Figure F.59 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision (avec CQ = 0) des connaissances relatives à l'application des actions

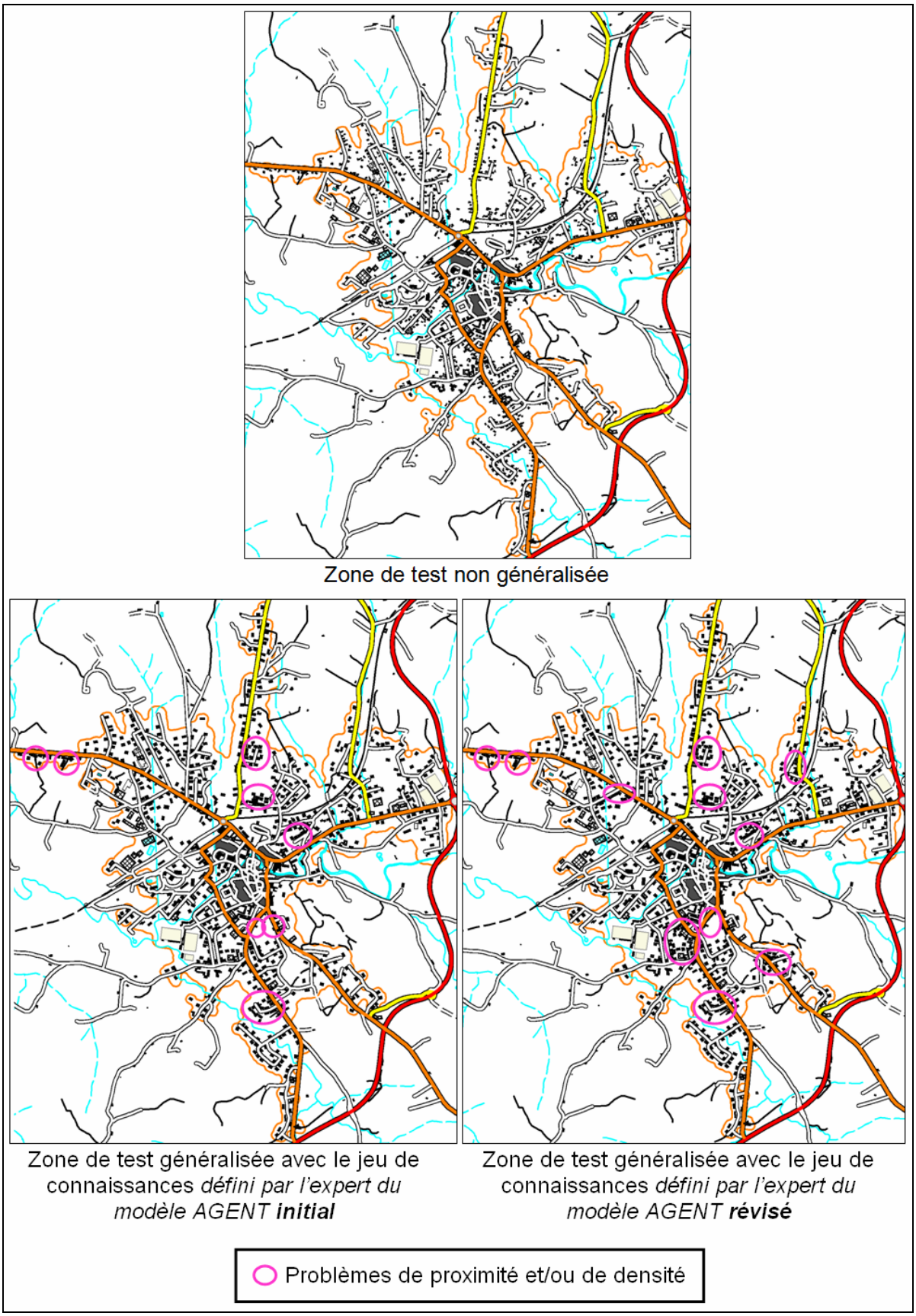


Figure F.60 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision (avec CQ = 0) des connaissances relatives à l'application des actions

Les tables suivantes donnent les résultats des différents jeux de connaissances obtenus après révision sur la zone de test :

Révision des connaissances relatives à l'application des actions pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.862**	0.355	0.761
Après révision avec ***CQ*** *= 0*	0.834	0.593	**0.786**
Après révision avec ***CQ*** *= 0.01*	0.849	0.420	0.763
Après révision avec ***CQ*** *= 0.1*	**0.862**	0.355	0.761

Révision des connaissances relatives à l'application des actions pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	0.807	0.69	0.784
Après révision avec ***CQ*** *= 0*	0.805	**0.72**	0.788
Après révision avec ***CQ*** *= 0.01*	0.815	0.693	**0.790**
Après révision avec ***CQ*** *= 0.1*	**0.822**	0.655	0.789

Révision des connaissances relatives à l'application des actions pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.838**	0.599	0.79
Après révision avec ***CQ*** *= 0*	0.808	**0.732**	0.792
Après révision avec ***CQ*** *= 0.01*	0.825	0.669	**0.794**
Après révision avec ***CQ*** *= 0.1*	**0.838**	0.599	0.79

Nous constatons que la révision des connaissances relatives à l'application des actions a bien permis d'améliorer les performances des trois jeux de connaissances. Les bases de règles révisées, en limitant les états pour lesquels les différentes actions ont été proposées, a permis au système d'augmenter son score d'efficience et en conséquence son score de performance.

F.III.7.3 Bilan de l'expérimentation

Cette expérimentation nous a permis de valider notre approche de révision des connaissances relatives à l'application des actions. Notre approche a en effet permis d'améliorer les trois jeux de connaissances.

Cette expérimentation nous a également permis de tester à nouveau nos diverses approches d'exploration. L'approche qui obtient cette fois les meilleurs résultats est l'approche agent. Cela s'explique par le fait qu'en considérant à chaque itération un voisinage plus large, l'approche agent a permis de ne pas rester bloqué au niveau de maxima locaux de la fonction

de performance. Cette prise en compte d'un voisinage très large peut être handicapante dans le cas où il y a beaucoup de partitions à traiter en même temps (comme c'était le cas pour les connaissances relatives à la priorité des contraintes) mais devient un avantage lorsque ce n'est pas le cas (comme ici pour les connaissances d'application des actions où l'approche de partitionnement a permis de limiter le nombre de partitions créées).

F.III.8 Révision de l'ensemble des connaissances

F.III.8.1 Contexte de l'expérimentation

Nous avons évoqué en partie D.II.2.1.1 le fait qu'il n'était pas judicieux de réviser indépendamment chaque type de connaissances sans tenir compte du résultat de la révision des autres types de connaissances. Nous avons proposé de réviser séquentiellement chaque type de connaissances en prenant en compte à chaque étape l'ensemble des connaissances déjà révisées.

Nous avons proposé dans ce cadre en partie D.II.2.1.2 un ordre spécifique de séquençage des différents processus de révision. Nous rappelons que cet ordre consiste à chercher dans un premier temps à converger le plus vite possible vers le meilleur état (mettre le meilleur état à gauche de l'arbre d'états) puis de chercher à élaguer l'arbre d'états.

Nous proposons ainsi de réviser en premier les connaissances relatives à la priorité des contraintes, puis de réviser les connaissances relatives à l'application des actions, puis celle relative à l'optimalité des états, celle relative à la fin de cycle, celles relatives aux restrictions d'application des actions et enfin celle relative à la validité des états.

Nous proposons dans cette partie F.III.8 de tester cet ordre et de le comparer à une révision indépendante des connaissances. Le jeu de connaissances obtenu par révision indépendante des connaissances correspond à l'ensemble des valeurs de connaissances obtenues après révision dans les parties précédentes.

F.III.8.2 Résultats obtenus

Les figures F.61, F.62 et F.63 présentent les trois jeux de connaissances obtenus après révision avec un coefficient de qualité des connaissances égal à 0 (prise en compte minimale des connaissances initiales). Nous constatons des similarités entre les jeux de connaissances obtenus par notre approche de révision séquentielle des connaissances et ceux obtenus par révision indépendante des connaissances. Nous remarquons des différences en particulier au niveau du critère de fin de cycle. En effet, avec notre révision séquentielle des connaissances, aucune règle de fin de cycle n'a été proposée par aucun des jeux de connaissances.

Concernant les points communs entre les trois jeux révisés, nous remarquons au niveau de l'ordre de traitements des conflits (la priorité des contraintes), que lorsque l'agent *groupement de bâtiments* souffre de graves problèmes de proximité, de satisfaction des sous-agents et de densité, les trois jeux de connaissances révisés proposent de traiter en priorité les problèmes de proximité, puis ceux de satisfaction des bâtiments, et enfin ceux de densité. On trouve d'ailleurs ici une similarité avec le jeu de connaissances *défini par l'expert en cartographie*.
Concernant les actions proposées, les trois jeux de connaissances proposent de ne jamais appliquer l'action de suppression/recentrage d'un bâtiment. Ils proposent par contre de toujours appliquer les quatre autres actions. La seule exception est le jeu de connaissances obtenu après révision du jeu de connaissances *défini par l'expert en cartographie*. Le jeu

propose l'action de suppression locale uniquement lorsque la contrainte de proximité est assez insatisfaite (règle similaire à celui du jeu de connaissances initial). Les valeurs de restriction définies pour chaque action sont très proches entre les trois jeux de connaissances révisés.

Pour conclure, les trois jeux de connaissances obtenus après révision semblent proches les uns des autres. Ces similarités peuvent s'expliquer par la valeur du *coefficient de qualité des connaissances* utilisée (*CK = 0*).
Les jeux révisés gardent néanmoins quelques spécificités de leur jeu de connaissances initial.

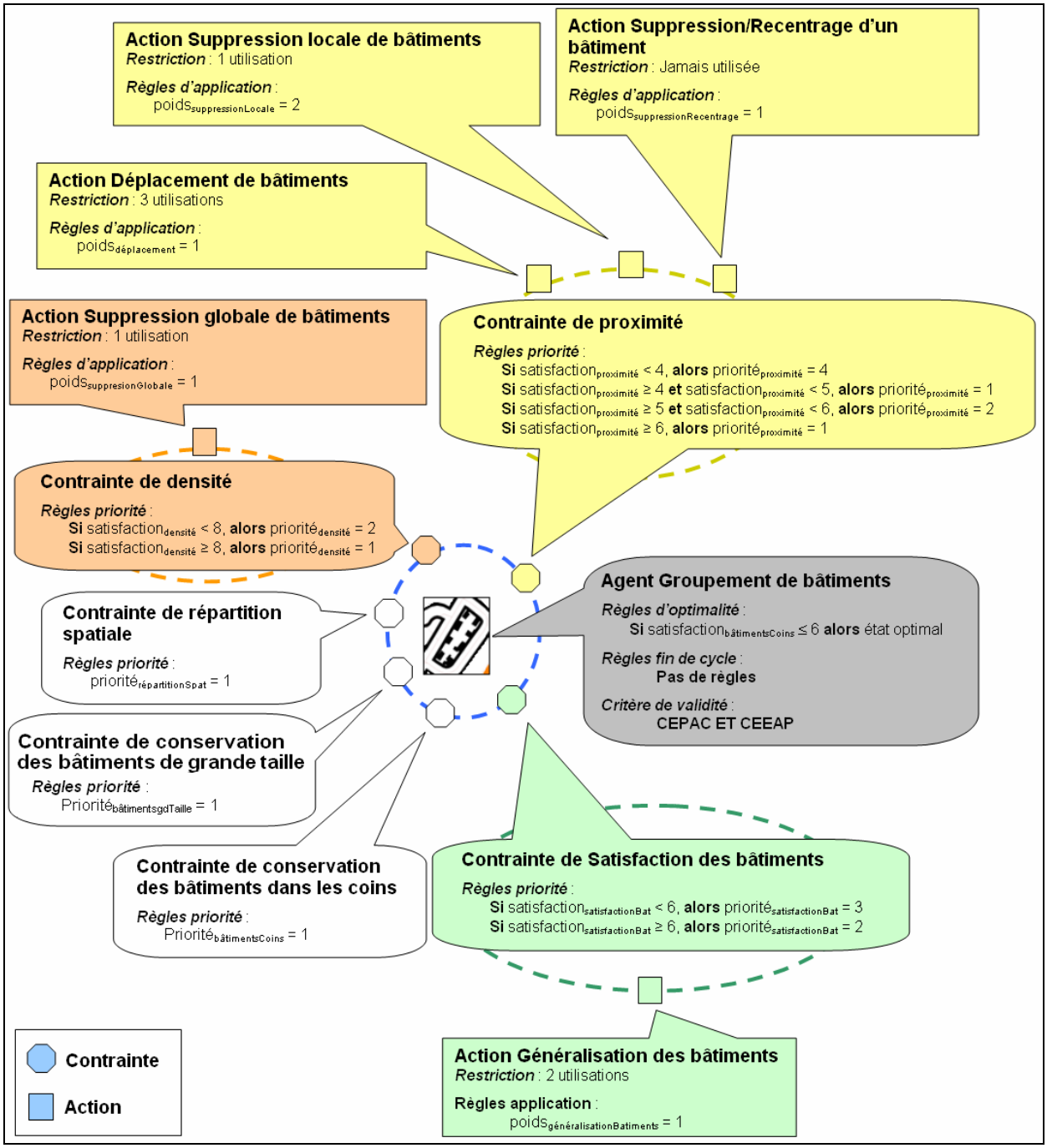


Figure F.61 Jeu de connaissances *basique* révisé (avec *CQ = 0*)

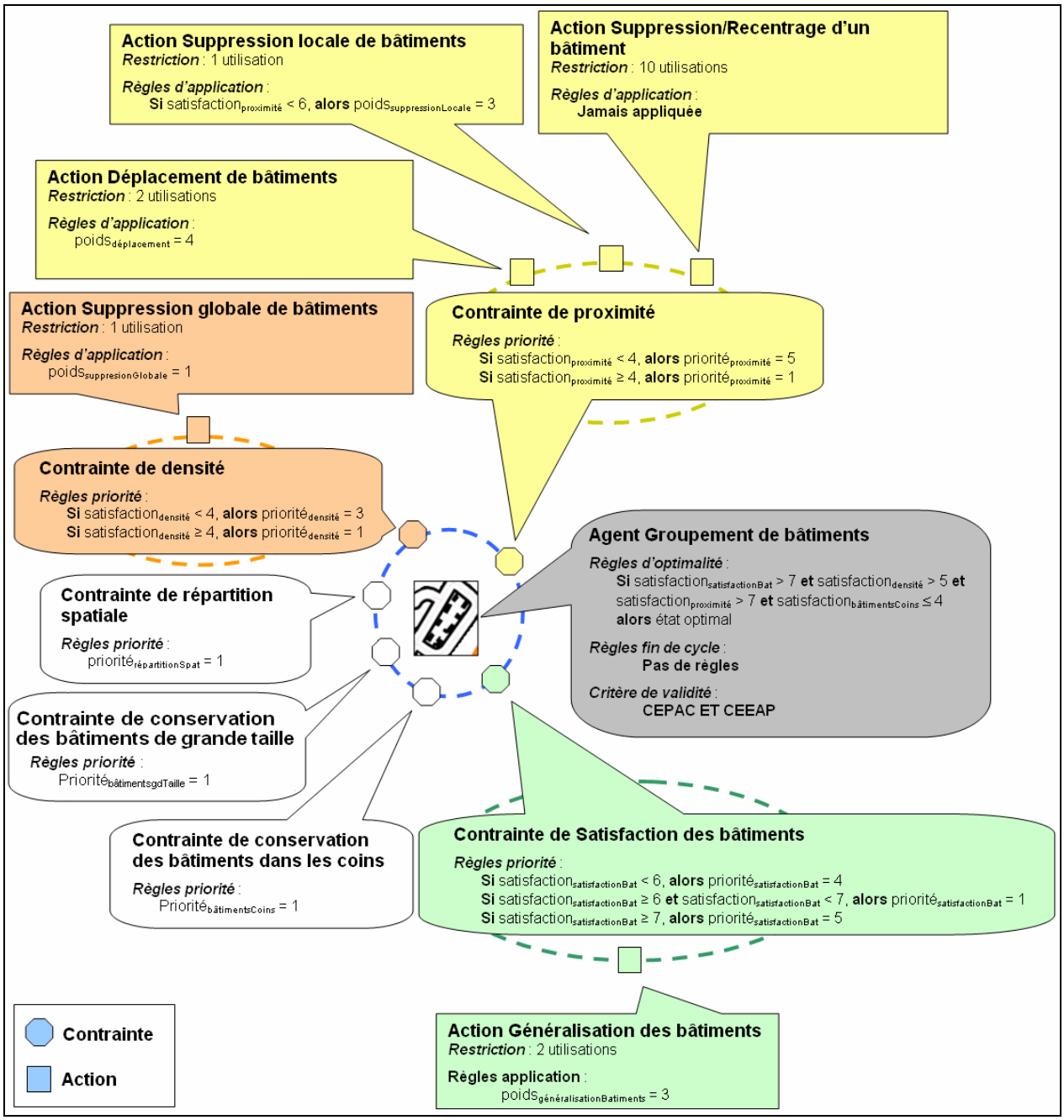


Figure F.62 Jeu de connaissances *défini par l'expert en cartographie* révisé (avec *CQ = 0*)

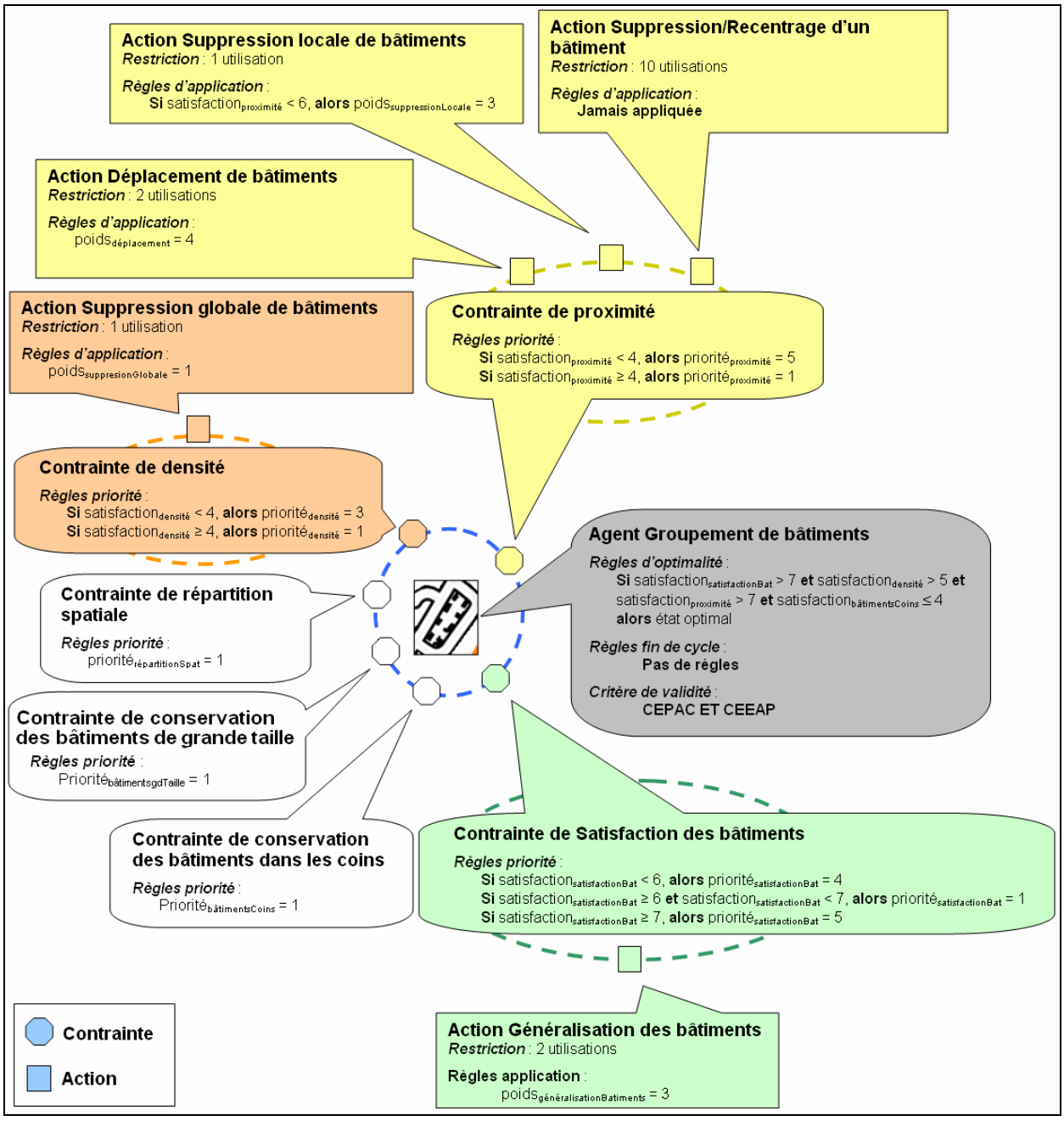


Figure F.63 Jeu de connaissances *défini par l'expert du modèle AGENT* révisé (avec *CQ = 0*)

Les figures F.64, F.65 et F.66 présentent les résultats obtenus avant et après révision sur la zone de test. Ces résultats montrent pour les trois jeux de connaissances un gain important d'efficience.

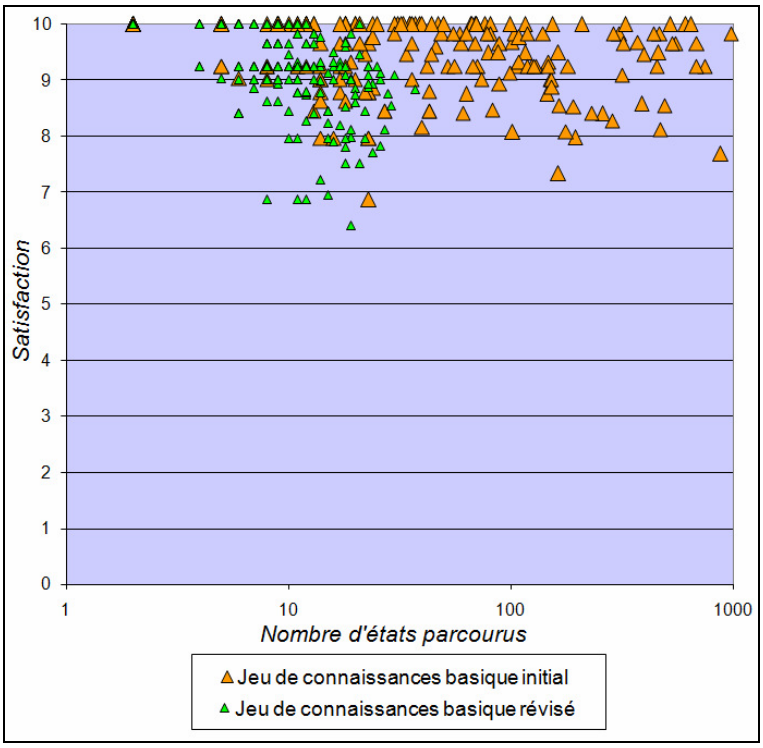


Figure F.64 Résultats obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision complète des connaissances (avec CQ = 0) (échelle logarithmique pour le nombre d'états parcourus)

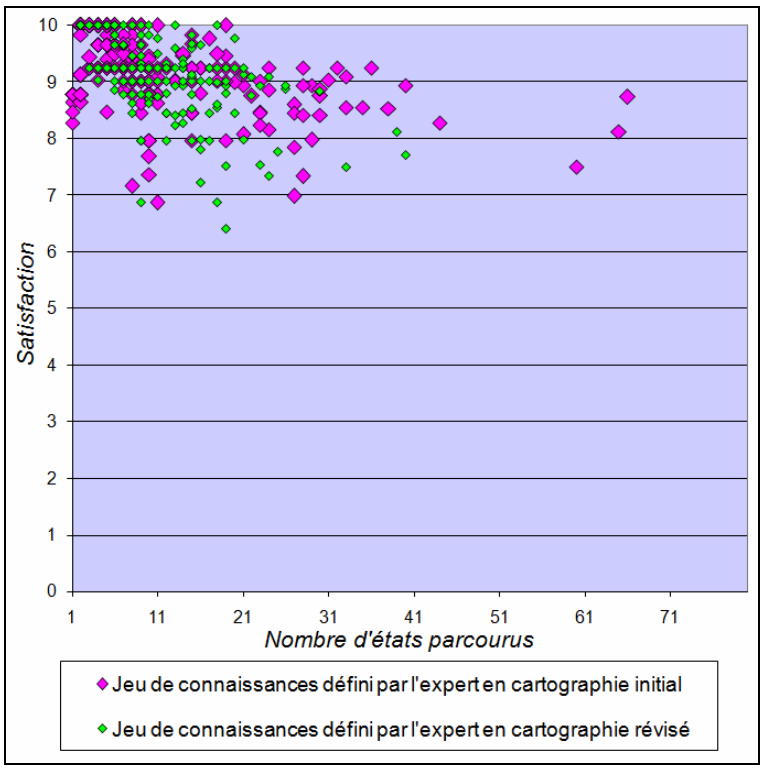


Figure F.65 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision complète des connaissances (avec *CQ = 0*)

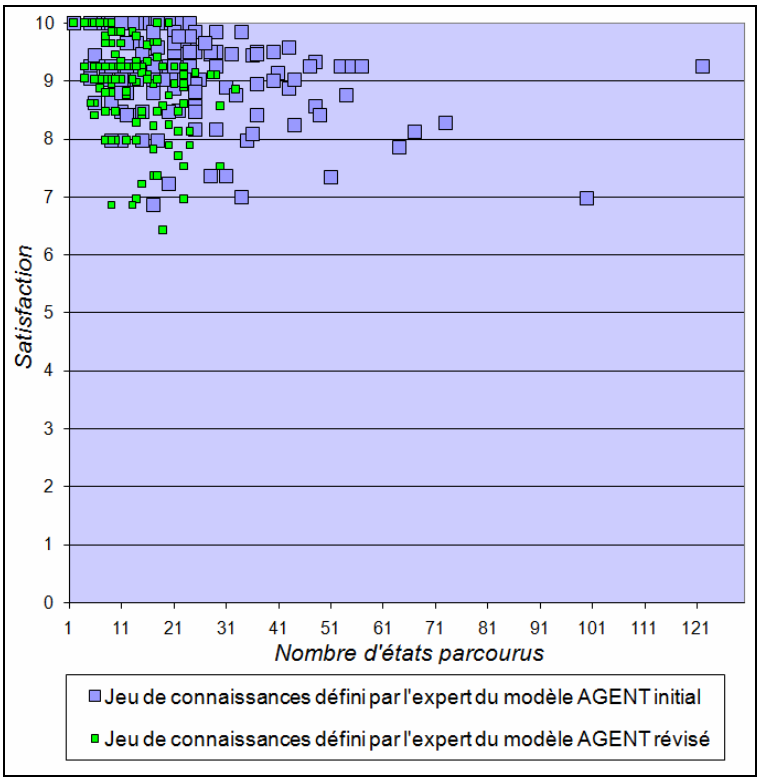


Figure F.66 Résultats obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision complète des connaissances (avec *CQ = 0*)

Les figures F.67, F.68 et F.69 présentent les résultats cartographiques obtenus avec ces jeux de connaissances révisés.
Les résultats obtenus avec les trois jeux de connaissances révisés sont très proches entre eux bien que l'on puisse toujours repérer, dans les résultats, certaines particularités propres aux différents jeux de connaissances initiaux révisés. Du point de vue de la qualité cartographique, les résultats obtenus avec les trois jeux révisés sont comparables aux résultats obtenus avec le jeu de connaissances *défini par l'expert du modèle AGENT* et sont donc très satisfaisants.

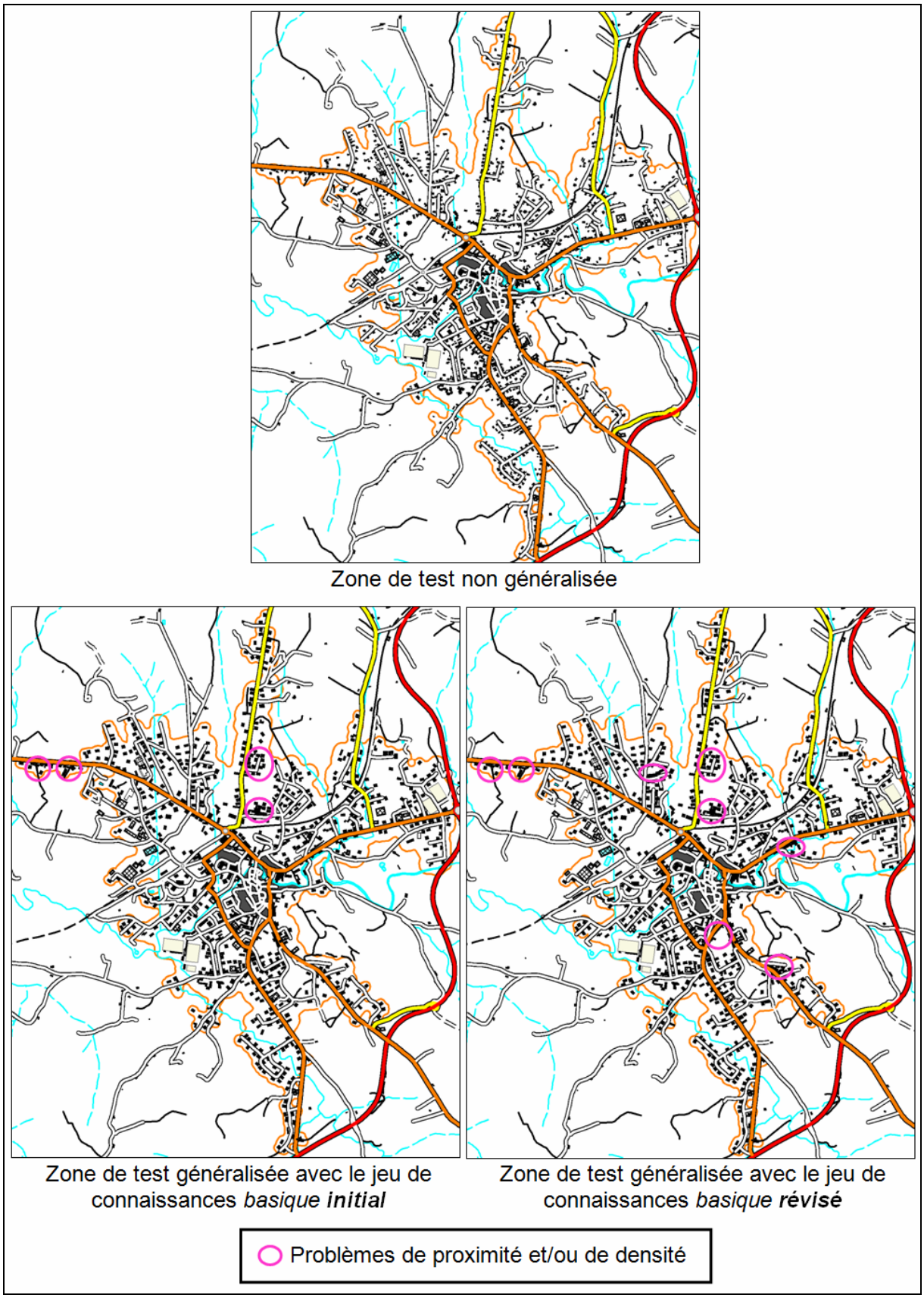


Figure F.67 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *basique* avant et après révision de l'ensemble des connaissances

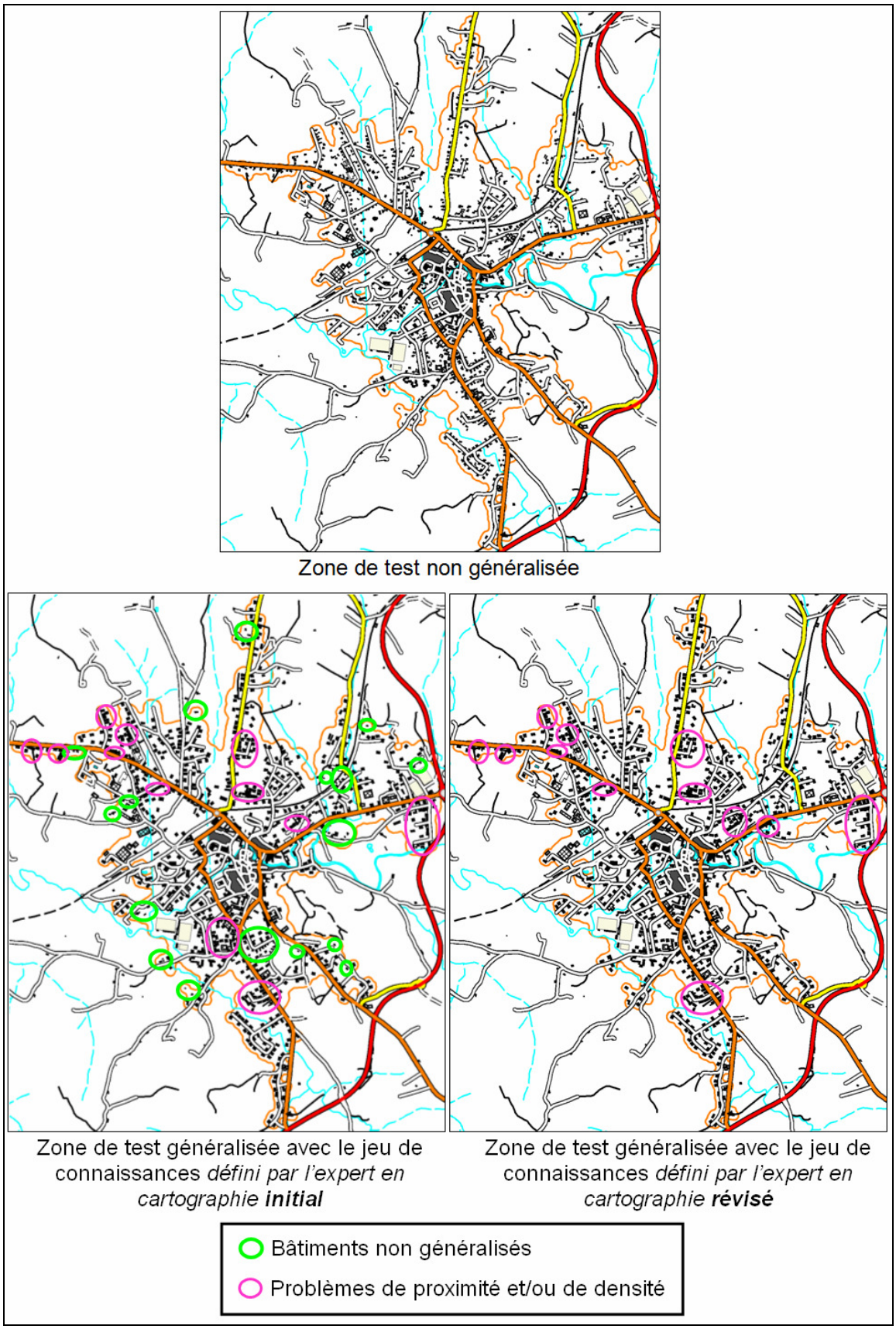


Figure F.68 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert en cartographie* avant et après révision de l'ensemble des connaissances

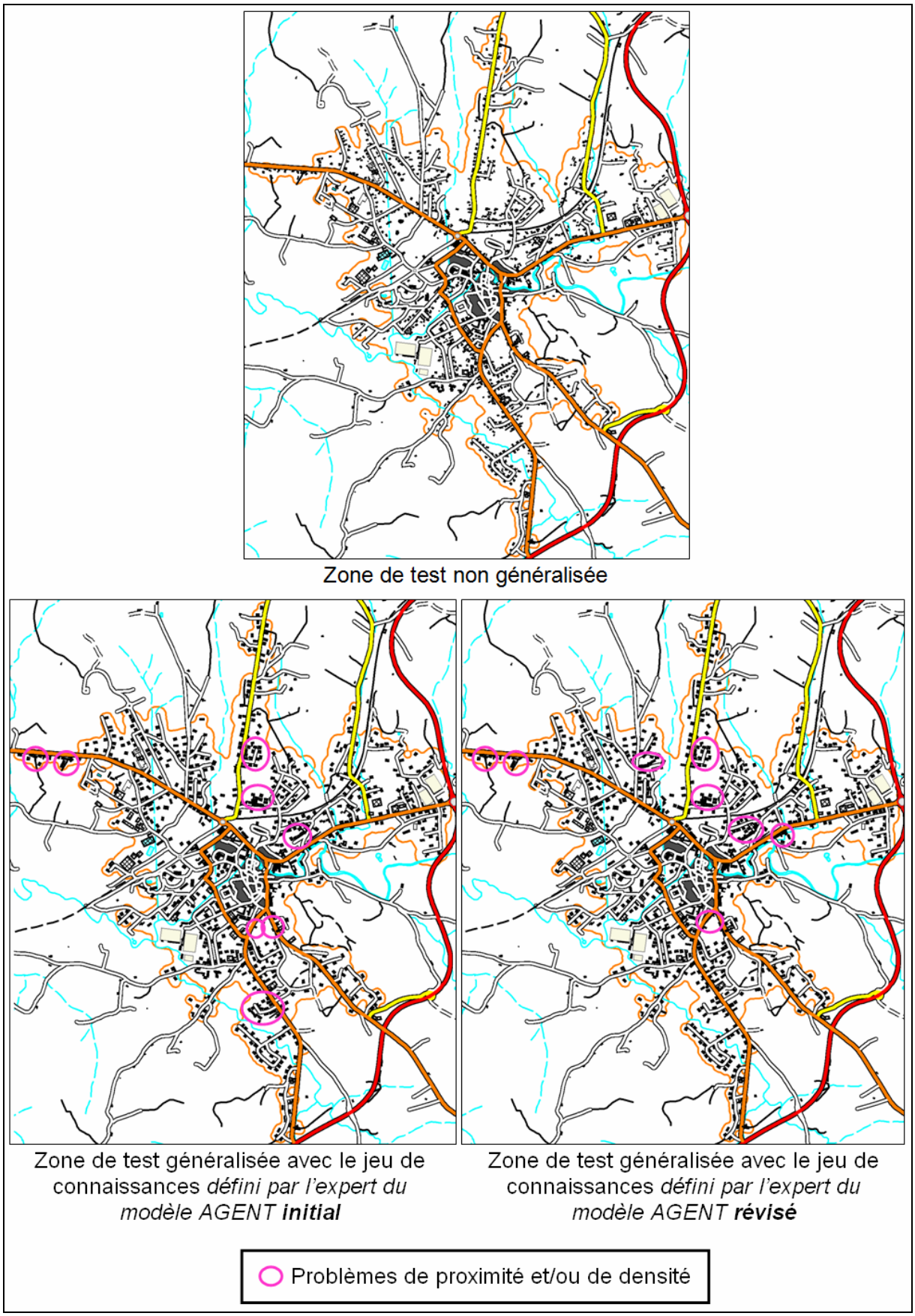


Figure F.69 Résultats cartographiques obtenus sur la zone de test par le jeu de connaissances *défini par l'expert du modèle AGENT* avant et après révision de l'ensemble des connaissances

Nous présentons dans les tables suivantes les résultats des différents jeux de connaissances obtenus après révision sur la zone de test. Nous donnons pour chacun d'eux, et pour trois valeurs du coefficient de qualité des connaissances, les résultats obtenus par révision indépendante de chacune des connaissances et par notre approche de révision séquentielle des connaissances.

Révision de l'ensemble des connaissances pour le jeu de connaissances *basique* :

		***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision		**0.862**	0.355	0.761
Révision indépendante de chaque connaissance	*Après révision avec **CQ** = 0*	0.804	0.701	0.784
	*Après révision avec **CQ** = 0.01*	0.813	0.662	0.783
	*Après révision avec **CQ** = 0.1*	0.817	0.642	0.782
Révision séquentielle des connaissances	*Après révision avec **CQ** = 0*	0.822	**0.712**	**0.799**
	*Après révision avec **CQ** = 0.01*	0.816	0.700	0.793
	*Après révision avec **CQ** = 0.1*	0.837	0.601	0.790

Révision de l'ensemble des connaissances pour le jeu de connaissances *défini par l'expert en cartographie* :

		***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision		0.807	0.69	0.784
Révision indépendante de chaque connaissance	*Après révision avec **CQ** = 0*	0.808	**0.744**	0.795
	*Après révision avec **CQ** = 0.01*	0.805	0.719	0.788
	*Après révision avec **CQ** = 0.1*	0.804	0.719	0.787
Révision séquentielle des connaissances	*Après révision avec **CQ** = 0*	0.823	0.712	0.801
	*Après révision avec **CQ** = 0.01*	**0.836**	0.69	**0.807**
	*Après révision avec **CQ** = 0.1*	0.822	0.652	0.788

Révision de l'ensemble des connaissances pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

		***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision		**0.838**	0.599	0.79
Révision indépendante de chaque connaissance	*Après révision avec **CQ** = 0*	0.787	**0.803**	0.79
	*Après révision avec **CQ** = 0.01*	0.804	0.762	0.796
	*Après révision avec **CQ** = 0.1*	0.807	0.733	0.792
Révision séquentielle des connaissances	*Après révision avec **CQ** = 0*	0.823	0.711	**0.8**
	*Après révision avec **CQ** = 0.01*	0.823	0.69	0.797
	*Après révision avec **CQ** = 0.1*	0.837	0.613	0.792

Ces résultats confirment les observations faites sur les nuages de points et sur les résultats cartographiques. La révision complète des connaissances par notre approche a permis d'améliorer significativement la qualité des résultats obtenus avec les trois jeux de connaissances.

Nous remarquons également, par comparaison avec une révision indépendante de chacune des connaissances, que notre approche de révision séquentielle a permis d'obtenir des résultats bien meilleurs. Ces résultats confirment les remarques faites en partie D.II.2.1.1. En effet, en raison des interdépendances existant entre les différents types de connaissances du modèle AGENT, une révision indépendante des connaissances pouvait entraîner de graves effets de bord. Ces effets de bord sont particulièrement visibles pour le jeu de connaissances *défini par l'expert du modèle AGENT* : alors que la révision de chaque connaissance (à part la connaissance relative à la fin de cycle) a permis d'améliorer cette connaissance, le jeu de connaissance rassemblant toutes les connaissances révisées a obtenu un score de performance égal à celui obtenu par le jeu de connaissances initial.

F.III.8.3 Influence du *coefficient de qualité des connaissances*

Nous constatons que les trois jeux obtenus après révision avec un *coefficient de qualité* égale à 0 (confiance très mauvaise dans les connaissances initiales) permettent d'obtenir tous les trois d'excellents scores de performances. Ces trois scores sont d'ailleurs très proches entre eux. Nous pouvons constater que le meilleur score de performance est obtenu par le jeu de connaissances résultant de la révision du jeu de connaissances *défini par l'expert en cartographie* avec un coefficient de qualité de 0.01. Nous expliquons ceci par le fait que l'expert en cartographie a défini de très bonnes règles pour certaines connaissances (les connaissances relatives à la priorité des contraintes, celles relatives à l'application de certaines actions) et de très mauvaises pour d'autres (connaissances relatives à la fin de cycle et à l'optimalité des états). Le coefficient de qualité a permis au processus de révision de conserver les règles de qualité définies par l'expert en cartographie même s'il était possible de trouver des règles qui étaient encore meilleures sur l'échantillon de révision (mais pas sur la

zone de test). En revanche, le processus de révision a supprimé les règles « *réellement* » mauvaises qui handicapaient les performances du système.
Dans le cas du jeu de connaissances défini par l'expert du modèle AGENT, toutes les règles définies étaient plutôt bonnes mais aucune n'était particulièrement excellente, le coefficient de qualité de 0.01 introduit pour toutes les connaissances n'a donc pas permis d'obtenir un jeu de connaissances encore meilleur.

Un détail qui peut paraître étonnant est le score de performance élevé obtenu par le jeu de connaissances résultant de la révision du jeu de connaissances *basique* avec un coefficient de qualité égal à 0.1. Ce score est même supérieur à celui obtenu par le jeu de connaissances résultant de la révision du jeu de connaissances *défini par l'expert en cartographie* avec un coefficient de qualité égale à 0.1 alors que le jeu de connaissances *défini par l'expert en cartographie* est meilleur que le jeu de connaissances *basique*. Nous expliquons cela par le fait que le jeu de connaissances *basique* ne comprend aucune règle intéressante mais ne comprend pas non plus de mauvaise règle. La raison pour laquelle celui-ci a obtenu un score de performance faible est le choix de son critère de validité. Lors du processus de révision, malgré la valeur élevée du *coefficient de qualité des connaissances*, le processus de révision a jugé pertinent de choisir un autre critère de validité assurant un compromis efficience/efficacité largement supérieur. Il a donc suffit au processus de révision de modifier ce critère de validité pour que les performances obtenues avec ce jeu de connaissances soient largement meilleures. Concernant le jeu de connaissances *défini par l'expert en cartographie*, celui-ci est handicapé par ses règles d'optimalité et de fin de cycle mal choisies. Malheureusement, en raison de la valeur élevée du *coefficient de qualité des connaissances*, le processus de révision n'a pas pu supprimer ces deux règles.
Un enseignement pouvant être tiré de cette analyse est qu'il ne faut surtout pas donner de valeurs élevées aux coefficients de qualité des connaissances introduisant un fort élagage de l'arbres d'états à moins d'être parfaitement sûr de ces valeurs.

Dans le cadre de cette expérimentation sur notre modèle complet de révision des connaissances, nous avons défini pour toutes les connaissances une même valeur pour leur coefficient de qualité. Or, l'intérêt d'un tel coefficient est de permettre à un expert de définir le degré de certitude sur la qualité de la valeur attribuée à chaque connaissance. Nous avons donc demandé à *l'expert du modèle AGENT* de définir, connaissance par connaissance, son degré de certitude.

Nous avons compilé dans le tableau suivant les degrés de certitude définis par *l'expert du modèle AGENT* :

Connaissance	**Degré de certitude** (1 = pas sûr du tout, 4 = certain)
Priorité de la contrainte de proximité	2
Priorité de la contrainte de densité	2
Priorité de la contrainte de satisfaction des bâtiments	2
Application de l'action de généralisation des bâtiments	4
Restriction sur l'application de l'action de généralisation des bâtiments	4
Application de l'action de déplacement de bâtiments	2

Restriction sur l'application de l'action de déplacement de bâtiments	1
Application de l'action de suppression locale de bâtiments	2
Restriction sur l'application de l'action de suppression locale de bâtiments	1
Application de l'action de suppression globale de bâtiments	2
Restriction sur l'application de l'action de suppression globale de bâtiments	4
Application de l'action de suppression/recentrage d'un bâtiment	2
Restriction sur l'application de l'action de suppression/recentrage d'un bâtiment	1
Optimalité des états	1
Fin de cycle	1
Validité des états	4

Nous avons donné les valeurs suivantes pour les coefficients de qualité des connaissances (à l'aide d'un étalonnage empirique):

- Degré de certitude = 1 $\Rightarrow$ CQ = 0
- Degré de certitude = 2 $\Rightarrow$ CQ = 0.01
- Degré de certitude = 3 $\Rightarrow$ CQ = 0.1
- Degré de certitude = 4 $\Rightarrow$ CQ = 1

Le score de performance obtenu sur le jeu de test par le jeu de connaissances résultant de la révision du jeu de connaissances *défini par l'expert du modèle AGENT* avec les *coefficients de qualité des connaissances* définis ci-dessus est égal à **0.807**.
Le jeu de connaissances révisé a donc obtenu un excellent score de performance. Ce score est supérieur à celui obtenu après révision avec un *coefficient de qualité des connaissances* égal à 0 pour l'ensemble des connaissances (valeur pour laquelle le processus de révision tient le moins compte des connaissances initiales). Cette dernière expérimentation démontre bien l'intérêt du coefficient de qualité des connaissances et de chercher à réviser des connaissances et non simplement à en acquérir de nouvelles.

F.III.8.4 Bilan de l'expérimentation

Cette expérimentation a permis de valider notre approche générale de révision des connaissances. En effet, les trois jeux de connaissances initiaux révisés par notre approche ont bien été améliorés de manière significative après révision.

Cette expérimentation a pu également montrer l'intérêt de réviser séquentiellement les connaissances en tenant compte des connaissances déjà révisées. Les résultats obtenus par cette approche sont bien meilleurs que ceux obtenus par révision indépendante de chacune des connaissances.

Enfin, nous avons pu montrer l'intérêt de la révision des connaissances par rapport à une simple acquisition de nouvelles. L'aspect révision est d'autant plus intéressant que l'on a introduit dans notre approche un *coefficient de qualité* permettant de jouer, pour chaque connaissance, sur le taux de modifications apportées par rapport au jeu de connaissances initial. Notre dernier test a permis de montrer qu'il était possible de jouer avec succès sur ce taux.

F.III.9 Bilan

Nous avons présenté au cours de cette partie F.III plusieurs expérimentations visant à évaluer notre approche générale de révision ainsi que diverses méthodes utilisées lors de à cette révision.

Nous avons pu constater au travers de l'ensemble de ces expérimentations que notre approche de révision était pertinente et qu'elle permettait bien d'améliorer la qualité de jeux de connaissances. Nous avons en particulier pu vérifier, par le biais d'expérimentations menées sur les connaissances relatives à l'optimalité des états, à la fin de cycle, à la priorité des contraintes et enfin à l'application des actions, que notre approche de *révision par analyse* destinée aux connaissances représentées sous forme de base de règles de $\mathcal{BR}$ (cf. D.IV.2) permettait de réviser avec succès les connaissances concernées par la révision.

Nous avons, au cours des tests menés sur la révision des connaissances relatives à l'optimalité des états et à la fin de cycle, testé les deux approches de partitionnement que nous avons proposées en partie D.IV.3.4. Les résultats de ces tests ont permis de montrer que l'approche par apprentissage et comparaison de règles donnait en général de meilleurs résultats et entraînait la construction de moins de partitions. Notre approche a par contre pour désavantage sa complexité algorithmique qui la rend inutilisable dès que le nombre de règles apprises est trop important (au-delà d'une quinzaine de règles).

Nous avons également pu valider au cours de ces tests notre approche de simplification de bases de règles. Celle-ci a en effet permis durant nos expérimentations de diminuer le nombre de règles obtenues après révision et donc d'améliorer la lisibilité des bases de règles ainsi que d'améliorer certaines bases de règles en les rendant plus génériques.

Nous avons également mené des tests sur les différentes approches d'exploration de l'espace des affectations de conclusion que nous avons proposées en partie D.IV.3.5. Ces tests ont montré que notre approche agent pouvait être plus efficace qu'une approche locale taboue dans le cadre de problèmes d'exploration pour lesquels le nombre *d'agents partition* n'était pas trop élevé. Ces expérimentations ont également montré que l'ajout d'un filtrage agent à un algorithme de recherche locale taboue pouvait permettre à cet algorithme de converger plus rapidement vers une meilleure solution.

Nos dernières expérimentations ont concerné notre approche générale de révision des connaissances basée sur la révision séquentielle de chaque type de connaissances. Ces expérimentations ont montré que notre approche était pertinente et qu'elle permettait d'améliorer significativement la qualité de jeux de connaissances. Ces expérimentations ont également permis de montrer que notre approche donnait de meilleurs résultats qu'une révision indépendante de l'ensemble des connaissances.

Nous avons testé pour l'ensemble des connaissances différentes valeurs pour le *coefficient de qualité des connaissances*. Nous avons ainsi constaté que celui-ci avait un rôle important dans le processus de révision du jeu de connaissances. Conformément à la définition de ce coefficient, nous avons pu constater que plus celui-ci avait une valeur élevée pour une connaissance, plus la valeur de celle-ci après révision était proche de sa valeur initiale. Nous avons également constaté que la définition, connaissance par connaissance, de la valeur de ce taux pouvait permettre d'obtenir après révision un jeu de connaissances encore plus performant. La définition de ce coefficient pour chaque connaissance peut néanmoins se

révéler délicate. En effet, estimer la qualité a priori des connaissances demande une grande maîtrise du modèle AGENT ainsi que des contraintes et des actions en jeu dans la généralisation de la classe d'agents géographiques considérée.

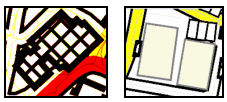
F.IV Expérimentation sur l'influence de l'échantillon de révision

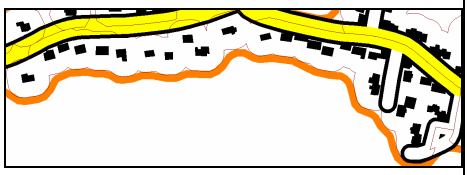
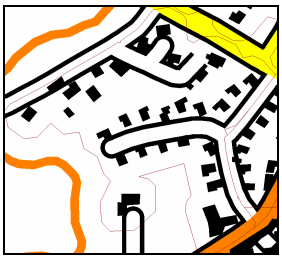
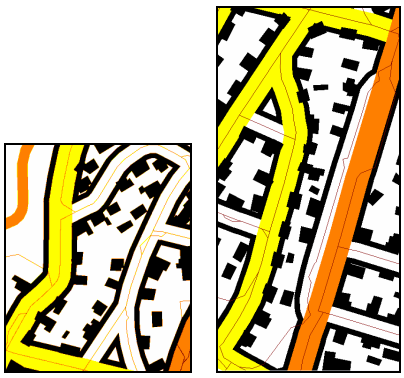
F.IV.1 Contexte de l'expérimentation

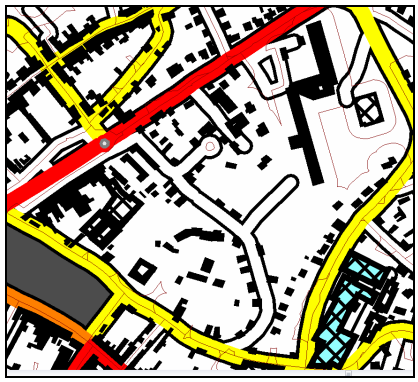
Nous avons mentionné en partie B.III que le choix de l'échantillon de révision pouvait avoir une influence sur la qualité des résultats. Nous avons proposé en partie C.II, une approche visant à sélectionner automatiquement un échantillon représentatif d'instances.Nous proposons de comparer les résultats de révision obtenus lorsque l'échantillon de révision a été sélectionné par notre approche, lorsqu'il a été sélectionné par un expert du modèle AGENT ou lorsqu'il a été tiré aléatoirement. Nous avons utilisé comme échantillon de révision pour toutes nos expérimentations 50 *groupements de bâtiments* sélectionnés dans la zone de révision.

F.IV.2 Résultats obtenus

Notre algorithme de classification non supervisée (*EM*) a divisé les 280 *groupements de bâtiments* de la zone de révision en 11 groupes que nous détaillons ci-dessous.

Groupe	Proportion	Type de groupements de bâtiments	Exemple de représentants du groupe
1	0.078	○ Très faible nombre de bâtiments ○ Peu ou pas de problèmes de proximité ○ Peu ou pas de problèmes de satisfaction des bâtiments	
2	0.14	○ Très faible nombre de bâtiments ○ Peu de problèmes de proximité moyens ○ Problèmes de satisfaction des bâtiments moyens	
3	0.117	○ Composé d'un faible nombre de bâtiments ○ Problèmes de proximité très importants	
4	0.102	○ Faible nombre de bâtiments ○ Composé de bâtiments importants ○ Problèmes de proximité importants ○ Problèmes de densité importants	
5	0.105	○ Nombre moyen de bâtiments ○ Problèmes de proximité moyens	
6	0.07	○ Nombre moyen de bâtiments ○ Présence de bâtiments dans les coins ○ Problèmes de proximité moyens ○ Problèmes de densité moyens	

7	0.059	○ Nombre moyen de bâtiments ○ Présence de bâtiments dans les coins ○ Problèmes de proximité très importants ○ Problèmes de densité très importants	
8	0.067	○ Grand nombre de bâtiments ○ Problèmes de proximité moyens ○ Problèmes de densité très faibles	 
9	0.129	○ Grand nombre de bâtiments ○ Problèmes de proximité importants ○ Problèmes de densité moyens	
10	0.109	○ Grand nombre de bâtiments ○ Composé de grands bâtiments ○ Problèmes de proximité très importants ○ Problèmes de densité moyens	 

11	0.024	○ Très grand nombre de bâtiments ○ Composé de grands bâtiments ○ Problèmes de proximité moyens ○ Problèmes de densité faibles	 

Le tableau suivant donne la répartition dans les groupes précédents des *groupements de bâtiments* des différents échantillons de révision. Nous remarquons que l'échantillon défini par l'expert comprend plus de *groupements de bâtiments* complexes (groupes 8, 9, 10 et 11) que l'échantillon défini automatiquement par notre approche. Le premier échantillon tiré aléatoirement admet une répartition au niveau des groupes proche de celle proposée par notre approche. Le second échantillon est plutôt composé de *groupements de bâtiments* complexes alors que le dernier échantillon est plutôt composé de *groupements de bâtiments* simples.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Echantillon sélectionné par notre approche	4	7	6	5	5	3	3	3	7	6	1
Echantillon sélectionné par l'expert	3	5	6	3	4	3	3	7	3	11	2
Echantillon tiré aléatoirement 1	5	3	6	7	4	3	5	2	8	5	2
Echantillon tiré aléatoirement 2	4	3	3	7	6	5	4	4	5	8	1
Echantillon tiré aléatoirement 3	7	9	8	2	3	3	2	4	8	4	0

Les tables suivantes donnent les résultats obtenus sur la zone de test avec les jeux de connaissances révisés à l'aide des différents échantillons de révision.

Révision de l'ensemble des connaissances pour le jeu de connaissances *basique* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.862**	0.355	0.761
Après révision avec l'échantillon de révision sélectionné par notre approche	0.822	0.712	**0.799**
Après révision avec l'échantillon de révision sélectionné par un expert	0.827	0.676	0.797
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 1	0.808	0.701	0.787
Après révision avec échantillon de révision sélectionné aléatoirement Echantillon 2	0.816	0.659	0.785
Après révision avec échantillon de révision sélectionné aléatoirement Echantillon 3	0.795	**0.79**	0.796

Révision de l'ensemble des connaissances pour le jeu de connaissances *défini par l'expert en cartographie* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	0.807	0.69	0.783
Après révision avec l'échantillon de révision sélectionné par notre approche	0.824	0.712	0.801
Après révision avec l'échantillon révision sélectionné par un expert	0.826	0.701	0.8
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 1	**0.841**	0.654	**0.803**
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 2	0.798	**0.781**	0.795
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 3	0.793	0.779	0.79

Révision de l'ensemble des connaissances pour le jeu de connaissances *défini par l'expert du modèle AGENT* :

	***Efficacité*(S_K,P_n)**	***Efficience*(S_K,P_n)**	***Perf*(S_K,P_n)**
Avant révision	**0.838**	0.599	0.79
Après révision avec l'échantillon de révision sélectionné par notre approche	0.823	0.711	**0.801**
Après révision avec l'échantillon de révision sélectionné par un expert	0.819	0.678	0.79
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 1	0.801	**0.754**	0.792
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 2	0.769	0.722	0.76
Après révision avec l'échantillon de révision sélectionné aléatoirement Echantillon 3	0.81	0.71	0.79

Ces résultats confirment que le choix de l'échantillon d'apprentissage a une influence et qu'il est important de choisir un échantillon représentatif de l'ensemble des instances traitées. Nous remarquons en effet que pour le jeu de connaissances défini par l'expert, la révision à partir de l'un des échantillons tirés aléatoirement (l'échantillon 2) a entraîné une détérioration des résultats obtenus sur la zone de test.

L'échantillon choisi par l'expert a permis d'obtenir dans l'ensemble de bons résultats mais inférieurs à ceux obtenus avec l'échantillon sélectionné automatiquement par notre approche. Parmi les trois échantillons d'apprentissage, seul le premier (proche de l'échantillon sélectionné par notre approche d'un point de vue répartition des *groupements de bâtiments* dans les groupes) a permis d'obtenir des résultats réellement meilleurs (avec amélioration significative des trois jeux de connaissances). Les deux autres échantillons tirés aléatoirement ont permis d'obtenir des résultats plus nuancés.

F.IV.3 Bilan de l'expérimentation

Cette expérimentation montre l'influence du choix de l'échantillon de révision dans la qualité des résultats obtenus après révision.

Elle permet de plus de vérifier la pertinence de notre approche de sélection automatique d'exemples. En effet, les résultats de révision obtenus avec l'échantillon sélectionné par notre approche sont meilleurs que les résultats obtenus après révision avec utilisation d'un échantillon défini par un expert.

Un point important de notre approche de sélection des exemples concerne le choix des mesures utilisées pour caractériser les instances du problème considéré. Si ce jeu de mesures n'est pas pertinent, l'échantillon choisi pourra ne pas être représentatif de l'ensemble des instances et donc ne pas garantir la qualité des connaissances pouvant être obtenues par révision à partir de celui-ci.

F.V Expérimentation de l'approche d'évaluation des jeux de mesures

F.V.1 Contexte de l'expérimentation

Nous avons montré dans les parties précédentes que notre approche de révision est efficace et qu'elle permet une amélioration significative de jeux de connaissances.

Nous nous intéressons dans cette partie F.V à la méthode d'évaluation de la qualité des jeux de mesures utilisées pour définir les connaissances (cf. D.III.3). Cette méthode se base sur une analyse de la cohérence de la base d'exemples pour en déduire une évaluation des jeux de mesures utilisés pour décrire les états des exemples de la base.

Nous proposons comme protocole d'expérimentation d'évaluer avec notre approche différents jeux de mesures de qualités a priori différentes et d'analyser les corrélations existant entre la qualité des jeux de mesures évaluée par notre approche et la qualité des résultats de généralisation obtenus avec un jeu de connaissances révisé en utilisant ces jeux de mesures. Nous rappelons que parmi les connaissances du modèle AGENT enrichi, seuls trois types dépendent d'un jeu de mesures : les connaissances relatives à l'application des actions, celle relative à l'optimalité des états et enfin celle relative à la fin de cycle. Nous évaluons les jeux de mesures utilisés pour définir ces trois types de connaissances. Nous évaluons, pour chaque type de connaissances, les performances obtenues après révision de ce type de connaissances (nous ne révisons qu'un type de connaissances à la fois). L'objectif est d'établir un lien entre la qualité des connaissances obtenues après révision et l'évaluation des jeux de mesures utilisés pour définir les connaissances.

Nous proposons enfin, pour chaque connaissance, de fournir la proportion d'exemples de la classe minoritaire de la base d'exemples associée à la connaissance. Nous rappelons que les bases d'exemples sont composées d'exemples exprimés dans un formalisme attributs/classe. Pour les connaissances relatives à l'application d'une action, la classe a deux valeurs possibles : « *appliquer* » ou « *ne pas appliquer* ». Pour la connaissance relative à l'optimalité des états, elle a pour valeur : « *optimal* » ou « *non optimal* ». Et pour la connaissance relative à la fin de cycle, elle a pour valeur : « *arrêter* » ou « *continuer* ». L'intérêt de cette information est qu'elle nous donne une indication sur la valeur maximale pouvant être atteinte par le taux d'incohérence. En effet, dans le cas où aucune mesure n'est pertinente, aucune des mesures n'est conservée après la phase de sélection des mesures pertinentes. Tous les exemples sont donc caractérisés de la même façon et le taux d'incohérence est ainsi égal à la proportion d'exemples de la classe minoritaire.

Pour ne pas biaiser la révision, nous proposons de partir du jeu de connaissances *basique*. Ce jeu n'introduit pas de connaissances initiales particulières et permet donc d'analyser plus facilement les jeux de connaissances obtenus après révision.

Nous présentons ci-dessous trois séries de jeux de mesures que nous proposons de tester :

- Des jeux de mesures « *non pertinents* » : ces jeux de mesures sont composés de mesures qui ne donnent aucune information sur les connaissances. Chaque jeu de mesures est ainsi composé d'une mesure renvoyant l'identifiant de l'agent géographique concerné et une autre renvoyant un nombre aléatoire.
- Des jeux de mesures « *satisfaction* » : ces jeux de mesures sont uniquement composés des satisfactions des contraintes. Ainsi, chaque jeu de mesures lié à une contrainte (et donc aussi à l'ensemble des actions proposées par cette contrainte) est composé uniquement de la satisfaction de la contrainte. Le jeu de mesures lié à l'agent *groupement de bâtiments* est composé de l'ensemble des satisfactions des contraintes de l'agent. L'intérêt de tester un tel jeu de mesures provient du fait que ce type de jeu de mesures est souvent utilisé dans le cadre de définition de connaissances de guidage. En effet, les satisfactions des contraintes sont en général des indicateurs intéressants sur l'état de l'agent géographique.
- Des jeux de mesures « *valeurs courantes* » : ces jeux de mesures correspondent à ceux que nous avons défini dans le cadre des expérimentations menées dans ce chapitre F (cf. F.II.1.5). Ils correspondent aux jeux de mesures « *satisfaction* » auxquels on a ajouté, pour chaque jeu de mesures lié une contrainte, l'ensemble des mesures utilisées par la contrainte pour calculer sa satisfaction (les valeurs courantes de la contrainte). Le jeu lié à l'agent géographique correspond à l'union de tous les jeux de mesures liés à l'ensemble des contraintes.

F.V.2 Résultats obtenus

Nous présentons pour chaque série de jeux de mesures les résultats sous la forme d'une table indiquant pour chaque connaissance :

- Les mesures utilisées par les règles obtenues après révision de la connaissance.
- Le score de performance obtenu par le système de généralisation sur la zone de test lorsque celui-ci utilise la base de règles obtenue après révision de la connaissance. Le score de performance obtenu par le jeu de connaissances initial est de 0.761.
  Notons que dans le cas des connaissances relatives à l'application des actions, l'ensemble de ces connaissances est révisé en même temps (un seul processus de révision est utilisé pour réviser l'ensemble des connaissances relatives à l'application des actions). Un seul score de performance sera donc donné pour l'ensemble de ces connaissances.
- La proportion d'exemples appartenant à la classe minoritaire dans la base d'exemples construite pour la connaissance.
- Le taux d'incohérence calculé par notre méthode.

La table suivante présente les résultats obtenus avec les jeux de mesures « *non pertinent* » :

	Mesures utilisées par la base de règles révisée	***Perf(S_K,P_n)***	***Proportion de la classe minoritaire***	***Incohérence***
Connaissance relative à la fin de cycle	Aucune	0.761	0.29	0.29
Connaissance relative à l'optimalité des états	Aucune	0.761	0.32	0.32
Connaissance relative à l'application de l'action de généralisation des bâtiments	Aucune	0.784	0.12	0.12
Connaissance relative à l'application de l'action de suppression locale de bâtiments	Aucune		0.16	0.16
Connaissance relative à l'application de l'action de déplacement de bâtiments	Aucune		0.41	0.41
Connaissance relative à l'application de l'action de suppression/recentrage d'un bâtiment	Aucune		0.04	0.04
Connaissance relative à l'application de l'action de suppression globale de bâtiments	Aucune		0.3	0.3

Nous constatons pour cette première série de jeux de mesures que la valeur d'incohérence est toujours égale à la proportion de la classe minoritaire, ce qui montre que les mesures n'apportent aucune information sur les connaissances. Ce constat est totalement cohérent avec la définition même de ces jeux de mesures. Nous remarquons également qu'aucune mesure n'est utilisée par les règles, ce qui démontre que notre approche de révision a bien permis de filtrer les mesures non pertinentes. Nous remarquons enfin que les performances après révision des connaissances relatives à l'application des actions ont augmenté alors qu'aucune mesure n'a été utilisée par les règles obtenues après révision. Nous expliquons cela par le fait que la règle « toujours appliquer l'action *suppression/recentrage d'un bâtiment* » a été remplacée après révision par la règle « ne jamais appliquer l'action *suppression/recentrage d'un bâtiment* » et que la règle « toujours appliquer l'action *suppression globale de bâtiments* » a été remplacée par la règle « ne jamais appliquer l'action *suppression globale de bâtiments* ». Comme nous l'avons vu pour la révision des connaissances relatives à la restriction des actions (F.III.3), ce type de règles permet d'améliorer l'efficience du système et ainsi ses performances.

La table suivante présente les résultats obtenus avec le jeu de mesures « *satisfaction* » :

	Mesures utilisées par les connaissances révisées	***Perf(S_K,P_n)***	***Proportion de la classe minoritaire***	***Incohérence***
Connaissance relative à la fin de cycle	Aucune	0.761	0.29	0.19
Connaissance relative à l'optimalité des états	$\text{satisfaction}_{\text{répartitionCoins}}$	0.763	0.32	0.14
Connaissance relative à l'application de l'action de généralisation des bâtiments	Aucune	0.784	0.12	0.12
Connaissance relative à l'application de l'action de suppression locale de bâtiments	Aucune		0.16	0.15
Connaissance relative à l'application de l'action de déplacement de bâtiments	Aucune		0.41	0.36
Connaissance relative à l'application de l'action de suppression/recentrage d'un bâtiment	Aucune		0.04	0.04
Connaissance relative à l'application de l'action de suppression globale de bâtiments	Aucune		0.3	0.3

Nous constatons pour cette seconde série de jeux de mesures que les taux d'incohérence sont restés très proches de la proportion de la classe minoritaire. Les satisfactions des contraintes n'apportent que peu d'informations supplémentaires sur les connaissances. Les seules connaissances pour lesquelles ces taux ont diminué significativement sont les connaissances relatives à la fin de cycle et à l'optimalité des états. C'est pour la connaissance relative à l'optimalité des états que le taux a le plus diminué et c'est aussi la seule connaissance pour laquelle des règles utilisant le jeu de mesures ont été définies. On constate donc une corrélation entre la baisse du taux d'incohérence et le fait qu'une mesure réellement pertinente ait été ajoutée au jeu de mesures (la satisfaction de la contrainte de *conservation des bâtiments dans les coins*).

Nous pouvons enfin déduire de cette seconde expérimentation que l'utilisation de jeux de mesures composés uniquement des satisfactions des contraintes est insuffisante pour la définition des connaissances.

La table suivante présente les résultats obtenus avec le jeu de mesures « *classique* » :

	Mesures utilisées par les connaissances révisées	***Perf(S_K,P_n)***	***Proportion de la classe minoritaire***	***Incohérence***
Connaissance relative à la fin de cycle	$satisfaction_{répartitionSpat}$ $satisfaction_{proximité}$ min_satisfactionBat	0.768	0.29	0.14
Connaissance relative à l'optimalité des états	$satisfaction_{RepartCoins}$	0.763	0.32	0.14
Connaissance relative à l'application de l'action de généralisation des bâtiments	Aucune	0.786	0.12	0.12
Connaissance relative à l'application de l'action de suppression locale de bâtiments	Aucune		0.16	0.15
Connaissance relative à l'application de l'action de déplacement de bâtiments	min_proximité		0.41	0.32
Connaissance relative à l'application de l'action de suppression/recentrage d'un bâtiment	Aucune		0.04	0.04
Connaissance relative à l'application de l'action de suppression globale de bâtiments	densité nb_bâtiments		0.3	0.15

Nous constatons pour cette dernière série de jeux de mesures que l'ajout de mesures pertinentes permet une diminution du taux d'incohérence d'un certain nombre de connaissances.

Pour les connaissances relatives à l'application de l'action de généralisation des bâtiments, pour celle relative à l'application de l'action de suppression locale de bâtiments et pour celle relative à l'application de l'action de suppression/recentrage d'un bâtiment, le taux d'incohérence est resté égal à la proportion d'exemples de la classe minoritaire. Cela indique que les nouvelles mesures n'ont pas apporté d'informations sur ces connaissances.

Pour les autres connaissances, le taux d'incohérence a diminué et des règles utilisant les mesures ont été définies. Nous constatons d'ailleurs que ces règles sont a priori pertinentes car elles ont permis une amélioration des performances du système de généralisation.
Cette dernière série de jeux de mesures, utilisée dans le cadre de l'ensemble des expérimentations menées dans ce chapitre F, s'est montrée dans l'ensemble assez pertinente. Néanmoins, il serait intéressant d'ajouter de nouvelles mesures pour les jeux de mesures dont le taux d'incohérence est élevé. Typiquement, l'ajout de nouvelles mesures pour le jeu de mesures lié à la contrainte de proximité pourrait s'avérer particulièrement rentable : c'est à partir de ce jeu que sont définies les règles de la connaissance relative à l'application de l'action de déplacement de bâtiments dont le taux d'incohérence est très élevé.

F.V.3 Bilan de l'expérimentation

Ces expérimentations ont permis de valider notre approche d'évaluation des jeux de mesures. En effet, nous avons diagnostiqué, à l'aide de notre approche d'évaluation, les jeux de mesures afin de révéler ceux étant non pertinents ou insuffisants pour la définition de connaissances de qualité.

Ces expérimentations montrent également que l'utilisation de jeux de mesures uniquement composés des satisfactions des contraintes pouvait s'avérer insuffisante pour définir des connaissances pertinentes. L'ajout de mesures supplémentaires est souvent indispensable.

Enfin, en analysant avec notre approche les jeux de mesures utilisés dans les expérimentations de ce chapitre, nous avons déterminé le jeu de mesures nécessitant le plus urgemment de mesures complémentaires.

F.VI Expérimentation du module de diagnostic

F.VI.1 Contexte de l'expérimentation

Nous proposons dans cette expérimentation de tester notre approche de diagnostic en ligne des connaissances.

Notre approche est basée sur une double analyse : une première locale, au niveau de chaque connaissance, et une seconde, globale, visant à caractériser la qualité des résultats obtenus après résolution d'une instance d'un problème (E.III.1). Ces analyses permettent d'évaluer différents critères qui sont utilisés pour caractériser la qualité de chacune des connaissances ainsi que l'efficacité du système de résolution de problèmes. Nous avons proposé d'utiliser une méthode de décision multicritère pour évaluer qualitativement, à partir des valeurs attribuées à chaque critère, la qualité globale des connaissances. Nous avons présenté deux méthodes de décision multicritère pouvant être employées pour notre problème : la méthode ELECTRE TRI et une méthode basée sur la théorie des fonctions de croyance. Nous testons ces deux méthodes de décision multicritère.

Nous proposons d'évaluer différents jeux de connaissances sur trois échantillons de *groupements de bâtiments* choisis aléatoirement sur notre zone de test. Le choix de prendre des échantillons tirés aléatoirement provient de notre volonté de se rapprocher de conditions réelles d'utilisation du système de généralisation. Les généralisations en « conditions réelles » concernent généralement un grand nombre d'agents géographiques. Or, l'ordre d'activation de ces agents est souvent défini aléatoirement.
Les trois échantillons considérés pour ces expérimentations sont chacun composés de 30 *groupements de bâtiments*. Ce nombre a été défini empiriquement : il permet au module de diagnostic de fournir une évaluation fiable de la qualité des connaissances et permet de fournir une première évaluation de la qualité des connaissances assez rapidement.

Nous testons la qualité de sept jeux de connaissances différents. Les trois premiers sont les jeux de connaissances initiaux présentés en partie F.II.2. Les trois suivants sont les jeux de connaissances obtenus après révision de ces trois jeux initiaux en partie F.III.8.2. Le dernier jeu est un jeu dans lequel aucune action n'est jamais proposée (jeu de connaissances « *sans action* »). L'intérêt d'évaluer un tel jeu est d'observer le comportement du module de diagnostic lorsqu'il est face à un jeu de connaissances dont l'efficience est maximale (comme aucune action n'est jamais proposée, le seul état visité lors d'une généralisation est l'état initial) et dont l'efficacité est mauvaise.

Nous rappelons que le module de diagnostic renvoie une catégorie de qualité pour le jeu de connaissances. Les catégories possibles sont les suivantes : *très mauvaises*, *mauvaises*, *moyennes*, *bonnes*, *très bonnes*.
Le système de diagnostic renvoie également un indice de robustesse (taux de confiance) de sa décision indiquant la fiabilité de celle-ci (noté *TC*). Nous rappelons que celui-ci est compris entre 0 et 1. Plus sa valeur est élevée, plus la décision est considérée comme fiable.

Les deux méthodes de décision multicritère ont été paramétrées à l'aide de tests réalisés sur la zone de révision à l'aide de jeux de connaissances différents de ceux testés dans cette expérimentation.

F.VI.2 Résultats obtenus

Le tableau suivant donne les catégories proposées par notre module de diagnostic pour chacun des jeux de connaissances et pour chaque échantillon de test :

		Méthode de décision multicritère : *ELECTRE TRI*	Méthode de décision multicritère : *Théorie des fonctions de croyances*
Jeu de connaissances *« sans action »*	Echantillon 1	**Mauvaises** *Avec TC = 1*	**Mauvaises** *Avec TC = 1*
	Echantillon 2	**Mauvaises** *Avec TC =1*	**Mauvaises** *Avec TC = 1*
	Echantillon 3	**Mauvaises** *Avec TC = 1*	**Mauvaises** *Avec TC = 1*
Jeu de connaissances *basique*	Echantillon 1	**Mauvaises** *Avec TC = 0.7*	**Moyennes** *Avec TC = 0.5*
	Echantillon 2	**Mauvaises** *Avec TC = 0.7*	**Moyennes** *Avec TC = 1*
	Echantillon 3	**Mauvaises** *Avec TC = 0.7*	**Moyennes** *Avec TC = 0.6*
Jeu de connaissances *défini par l'expert en cartographie*	Echantillon 1	**Mauvaises** *Avec TC = 0.8*	**Moyennes** *Avec TC = 1*
	Echantillon 2	**Mauvaises** *Avec TC = 0.8*	**Moyennes** *Avec TC = 1*
	Echantillon 3	**Mauvaises** *Avec TC = 0.7*	**Moyennes** *Avec TC = 1*
Jeu de connaissances *défini par l'expert du modèle AGENT*	Echantillon 1	**Moyennes** *Avec TC = 0.4*	**Bonnes** *Avec TC = 1*
	Echantillon 2	**Mauvaises** *Avec TC = 0.6*	**Moyennes** *Avec TC = 1*
	Echantillon 3	**Moyennes** *Avec TC = 0.6*	**Moyennes** *Avec TC = 0.5*
Jeu de connaissances *basique* **révisé**	Echantillon 1	**Bonnes** *Avec TC = 0.4*	**Bonnes** *Avec TC = 1*
	Echantillon 2	**Bonnes** *Avec TC = 0.6*	**Bonnes** *Avec TC = 1*
	Echantillon 3	**Bonnes** *Avec TC = 0.6*	**Bonnes** *Avec TC = 0.5*
Jeu de connaissances *défini par l'expert en*	Echantillon 1	**Bonnes** *Avec TC = 1*	**Bonnes** *Avec TC = 0.9*
	Echantillon 2	**Bonnes** *Avec TC = 1*	**Bonnes** *Avec TC = 1*

cartographie **révisé**	Echantillon 3	**Bonnes** *Avec TC = 0.6*	**Bonnes** *Avec TC = 0.5*
Jeu de connaissances *défini par l'expert du modèle AGENT* **révisé**	Echantillon 1	**Bonnes** *Avec TC = 0.4*	**Bonnes** *Avec TC = 1*
	Echantillon 2	**Bonnes** *Avec TC = 1*	**Bonnes** *Avec TC = 1*
	Echantillon 3	**Bonnes** *Avec TC = 0.6*	**Bonnes** *Avec TC = 0.5*

Nous remarquons que les catégories proposées par notre approche de diagnostic sont cohérentes avec la qualité des jeux de connaissances pour les deux méthodes de décision multicritère. Ainsi, les jeux de connaissances révisés ont toujours été classés comme étant de catégorie « *bonnes* » alors que les autres jeux ont été classés en catégorie « *moyennes* » ou « *mauvaises* ». Concernant les taux de confiance, nous remarquons que leur valeur varie entre les différents échantillons pour la plupart des jeux de connaissances testés. Ce constat illustre l'intérêt de notre approche de test de variation des valeurs des paramètres et de vote majoritaire. En effet, cette approche permet de garder des décisions cohérentes malgré les différences entre les trois échantillons de test.

Un point important est que le système de diagnostic a bien permis d'identifier le jeu de connaissances ne proposant pas d'action comme étant de catégorie « *mauvaises* » et cela pour les deux méthodes de décision multicritère. Ce jeu de connaissances n'a pas été classé en catégorie « *très mauvaises* » car la satisfaction d'un agent géographique même non généralisé est rarement égale à 1 mais plutôt proche de 7 (cf. F.II.4.2).

Un autre point important est que les catégories proposées pour un même jeu de connaissances sont cohérentes pour les trois échantillons. Les seules exceptions à ce niveau concernent le jeu de connaissances défini par l'expert, où le module de diagnostic a, avec chacune des deux méthodes de décision multicritère, proposé pour l'un des échantillons une catégorie différente de celle proposée pour les deux autres échantillons.

Nous remarquons que les évaluations données avec la méthode ELECTRE TRI sont globalement plus sévères que celles données avec la méthode issue de la théorie des fonctions de croyances. Cet écart provient du paramétrage des deux méthodes : l'utilisation de l'une et l'autre de ces méthodes implique de définir de nombreux paramètres. Il n'est pas possible de faire un lien direct entre les paramètres de la méthode ELECTRE TRI et ceux de la méthode basée sur les fonctions de croyance. Nous n'avons donc pas pu directement fournir un même jeu de paramètres pour les deux méthodes. Lors de la phase de définition des paramètres, nous n'avons pas cherché à harmoniser les résultats obtenus avec chacune des deux méthodes mais juste à assurer la cohérence des résultats par rapport aux jeux de connaissances testés.

F.VI.3 Bilan de l'expérimentation

Cette expérimentation a permis de vérifier la pertinence de notre approche de diagnostic en ligne de la qualité de jeux de connaissances. Nous avons montré que notre approche de diagnostic permettait d'évaluer avec pertinence un ensemble de jeux de connaissances.

Les deux méthodes de décision multicritère se sont révélées toutes les deux adaptées à notre problème de diagnostic de qualité de jeux de connaissances. Un point clef de ces deux

méthodes concerne leur paramétrage. En effet, ces deux méthodes nécessitent de définir un nombre important de paramètres qui ont une influence directe sur la qualité des résultats. De ce point de vue, la méthode issue de la théorie des fonctions de croyance comprend moins de paramètres à définir. Toutefois, ces derniers peuvent se révéler plus difficiles à définir que ceux de la méthode ELETRE TRI.

F.VII Discussion sur les résultats

F.VII.1 Evaluation des résultats

Nous avons présenté dans les parties précédentes un ensemble d'expérimentations visant à évaluer l'ensemble des approches et des méthodes que nous avons présentées dans de le cadre de ce travail de thèse. Nous avons présenté, en partie F.III, une série d'expérimentations qui ont permis de vérifier la validité notre approche générale de révision des connaissances. Nous avons révisé dans le cadre de cette série d'expérimentations trois jeux de connaissances de différentes qualités.

L'intérêt de partir de profils différents était double : d'une part, l'enjeu était de déterminer s'il était possible d'améliorer un jeu de connaissances dont la qualité était déjà bonne avant révision (le jeu de connaissances *défini par l'expert du modèle AGENT*) et d'autre part de pouvoir analyser l'influence du jeu de connaissances initial sur la qualité du jeu de connaissances final.

Concernant le premier point, ces expérimentations ont montré que notre approche permettait aussi bien d'améliorer des jeux de connaissances de qualité moyenne (le jeu de connaissances *basique*) que des jeux de connaissances de bonne qualité (les deux jeux de connaissances *définis par les experts*).

En ce qui concerne l'influence du jeu de connaissances initial sur la qualité du jeu de connaissances obtenu après révision, celle-ci dépend pour beaucoup des valeurs données par l'utilisateur aux *coefficients de qualité des connaissances*. Ainsi, nous avons pu observer que les trois jeux obtenus après révision avec un coefficient de qualité des connaissances égal à 0 avaient de nombreux points communs entre eux même s'ils gardaient également certaines spécificités de leur jeu d'origine. Nous avons également pu confirmer que l'augmentation de la valeur du *coefficient de qualité des connaissances* se traduisait bien par l'obtention après révision d'un jeu de connaissances plus proche du jeu d'origine.
Le dernier test mené nous a permis de constater qu'il était possible, en partant d'un jeu de connaissances de qualité et en définissant pertinemment la valeur du coefficient de qualité de chaque connaissance, d'obtenir après révision un jeu de connaissances de qualité supérieure à celui obtenu avec une valeur égale à 0 pour l'ensemble des coefficients de qualité des connaissances. Nous expliquons cela par les imperfections de l'échantillon de révision. En effet, il est en pratique impossible d'obtenir un échantillon de révision parfaitement représentatif de l'ensemble des instances du problème traité. La révision est toujours biaisée pour un échantillon particulier. L'expert peut avoir, pour certaines connaissances, des informations supplémentaires peu visibles sur l'échantillon de révision. C'est d'ailleurs là que réside le principal intérêt de réviser les connaissances plutôt que de chercher à en acquérir de nouvelles. Cet aspect révision sera d'autant plus important que les jeux de connaissances initiaux seront de qualité et que l'expert sera capable de définir avec certitude quelles connaissances doivent être conservées.

F.VII.2 Difficultés posées

L'ensemble des expérimentations menées dans ce chapitre F a permis de démontrer que la mise en œuvre de notre approche permet de réviser avec succès des jeux de connaissances. Nous proposons dans cette partie F.VII.2 de soulever les difficultés posées par la mise en œuvre de notre approche de révision dans le cadre d'un problème donné.

F.VII.2.1 Evaluation des performances

La principale difficulté concerne l'évaluation des performances du système de résolution de problèmes. En effet, notre approche de révision, permet de réviser un jeu de connaissances de façon totalement automatique. Elle est basée sur l'évaluation des performances du système sur un échantillon d'instances du problème. Ainsi, pouvoir assurer que la fonction utilisée pour évaluer les performances du système correspond parfaitement aux besoins de l'utilisateur et que l'échantillon d'instances utilisé pour la révision est parfaitement représentatif de l'ensemble des instances que l'utilisateur souhaite traiter permet de garantir qu'un jeu de connaissances révisé par notre approche sera de qualité égale ou supérieure au jeu de connaissances initial. En pratique, ces deux conditions sont très difficiles à vérifier.

La définition d'une fonction représentative des besoins de l'utilisateur est un problème particulièrement complexe. Cette fonction doit, en fonction des besoins de l'utilisateur, tenir compte de l'efficacité du système ainsi que de son efficience.

Concernant son efficacité, un premier point à aborder est celui de la définition d'une fonction pertinente d'évaluation des états de l'entité considérée (la fonction Q_P, cf. B.III.1.1). Définir une telle fonction peut soulever de nombreuses difficultés.

Ainsi, même dans le cadre du modèle AGENT qui a été au cœur de nombreux travaux et qui est aujourd'hui utilisé pour la production de carte à l'IGN, la question de l'évaluation d'un état d'un agent géographique (sa satisfaction) reste encore posée. La fonction que nous avons utilisée dans le cadre de notre expérimentation est celle implémentée dans le logiciel Clarity™. Elle consiste en une moyenne des satisfactions des contraintes pondérée par leur importance. Comme évoqué en partie F.IV.2, cette fonction a tendance, en pratique, à n'utiliser qu'une plage limitée de valeurs. De plus, elle ne permet pas de faire de différence entre un état où toutes les satisfactions ont une valeur homogène entre elles et un état où ces valeurs sont très disparates (typiquement le cas où toutes les contraintes sont parfaitement satisfaites sauf une qui est très insatisfaite). Peu de travaux se sont intéressés à l'évaluation automatique des résultats d'une généralisation. Nous pouvons néanmoins citer dans le domaine les travaux de (Bard, 2004) qui proposent une méthode basée sur la définition de fonctions d'évaluation de caractéristiques et sur leur agrégation au sein d'une fonction d'évaluation globale. Une autre voie encore peu exploitée pour l'évaluation est d'utiliser l'apprentissage artificiel pour apprendre une fonction d'évaluation à partir des satisfactions des contraintes. Une méthodologie, proche de celle employée par (Ruas & Holzapfel, 2003) pour caractériser les alignements de bâtiments, peut être employée. Ainsi une fonction d'évaluation pourrait être apprise par analyse d'échantillons de généralisation notés par des experts.

Disposer d'une fonction pertinente d'évaluation de la qualité d'un état et donc du résultat de la résolution d'une instance du problème considéré est un premier point important pour évaluer l'efficacité d'un système de résolution de problèmes mais n'est pas suffisant. En effet, une seconde question concerne la définition de comment, à partir de deux résolutions

différentes d'un même ensemble d'instances, déterminer laquelle des deux résolutions est la meilleure. De nombreux critères peuvent être utilisés pour choisir une résolution plutôt qu'une autre : la moyenne des qualités de chaque instance, la médiane, le premier quartile, le minimum, etc. Nous nous sommes contentés pour nos expérimentations de faire une moyenne entre la moyenne des qualités individuelles et le premier quartile. Une fonction plus complexe aurait sans doute permis une meilleure adéquation entre les résultats fournis par notre fonction d'évaluation de l'efficacité et ceux que pourrait donner un cartographe. Encore une fois, l'utilisation d'une approche par apprentissage artificiel est envisageable pour apprendre une « *bonne* » fonction d'évaluation de la qualité.

Définir un jeu de connaissances maximisant l'efficacité du système peut être simple pour certains systèmes de résolution de problèmes. Ainsi, pour le modèle AGENT, nous pouvons garantir que le jeu de connaissances *basique* fournira toujours, pour chaque agent *groupement de bâtiments* généralisé, le meilleur état possible (ou l'un des meilleurs s'il y en a plusieurs) du point de vue de la satisfaction. Malheureusement, un tel jeu de connaissances est inutilisable en pratique pour la généralisation d'un nombre important d'agents *groupement de bâtiments*. En effet, le nombre d'états parcourus pour la généralisation est tel que le temps nécessaire à la généralisation de l'ensemble des *groupements de bâtiments* d'une ville, même petite, est trop long. Il n'est pas possible de prendre en considération uniquement l'efficacité du système de généralisation. Il est nécessaire de tenir un minimum compte de l'efficience du système.

Nous avons proposé pour l'efficience une fonction complexe tenant compte de la moyenne du nombre d'états parcourus pour la généralisation de chaque *groupement de bâtiments* pondérée par le nombre de bâtiments constituant le groupement. S'il est simple de définir des critères pour quantifier l'efficience (le nombre d'états parcourus, le temps nécessaire à la généralisation de l'ensemble des groupements de bâtiments, etc.), il est bien plus complexe de définir une échelle pour cette efficience afin de pouvoir la comparer directement à l'efficacité. En effet, nous cherchons avant tout à trouver un compromis entre efficience et efficacité, or il est extrêmement complexe de définir par des formules en quoi une légère baisse de l'efficacité peut être acceptée si l'efficience augmente significativement. Cette question est d'autant plus complexe que ce compromis à trouver entre efficience et efficacité est difficilement quantifiable par un facteur fixe. En effet, dans le cas où l'efficience est déjà très bonne, il ne sert peut être à rien de chercher à gagner encore en efficience, alors que dans le cas où l'efficience est vraiment très mauvaise, il sera peut être alors nécessaire de faire de lourds sacrifices en termes d'efficacité pour rendre l'efficience acceptable.

F.VII.2.2 Représentation de l'échantillon de révision

Il est ainsi important de définir une fonction d'évaluation des performances permettant d'exprimer pertinemment les besoins de l'utilisateur. Un second point important pour assurer le succès de la révision d'un jeu de connaissances par notre approche est la représentativité de l'échantillon d'instances utilisé pour réviser les connaissances. Nous avons déjà évoqué ce problème en partie B.IV.2 et nous avons proposé dans ce cadre une approche permettant de sélectionner parmi un ensemble d'instances, un ensemble a priori représentatif (cf. C.II.2). Nous avons montré que, dans le cadre des expérimentations menées en partie F.IV, notre approche pouvait permettre de sélectionner un échantillon d'instances pertinent. Le succès de cette approche est avant tout conditionné par le jeu de mesures utilisé. Ainsi, si nous avions utilisé un jeu de mesures non pertinent pour caractériser les agents *groupement de bâtiments*, l'échantillon de *groupements de bâtiments* sélectionné aurait pu n'être absolument pas

représentatif de l'ensemble des *groupements de bâtiments*. Le choix du jeu de mesures pour décrire les instances du problème considéré dans le cadre de notre approche de sélection de l'échantillon de révision pourra donc avoir une influence majeure sur la qualité du jeu de connaissances obtenu après révision.

F.VII.2.3 Pertinence des jeux de mesures

Le choix d'un jeu de mesures intervient également dans la définition de certaines connaissances. Ainsi, dans le cadre du modèle AGENT, les connaissances relatives à l'application des actions, celle relative à l'optimalité des états et celle relative à la fin de cycle, sont définies chacune à partir d'un jeu de mesures. Comme nous l'avons montré en partie F.V, le choix de ce jeu de mesures est très important pour la qualité des connaissances pouvant être définies à partir de celui-ci. Nous avons proposé dans ce cadre une approche en partie C.IV visant à évaluer les jeux de mesures testés dans ce chapitre. Cette approche permet de détecter les lacunes des jeux de mesures et peut aider le travail de l'expert dans sa définition de jeux de mesures pertinents.

F.VI Bilan

Nous avons proposé dans ce dernier chapitre une série d'expérimentations qui ont permis d'évaluer l'ensemble des méthodes et des approches que nous avons proposées dans ce travail de thèse. Nous avons ainsi testé l'ensemble de nos approches de révision et de diagnostic dans le cadre de la révision des connaissances du modèle AGENT pour le problème de la généralisation au 1 : 50 000 des agents *groupement de bâtiments*.

Après avoir introduit le contexte de nos expérimentations en partie F.II, nous avons présenté en partie F.III une série d'expérimentations visant à évaluer notre approche générale de révision des connaissances ainsi que l'ensemble des méthodes présentées au cours de la phase d'analyse (cf. D). Nous avons ainsi pu montrer que notre approche générale de révision permettait bien d'améliorer les performances des jeux de connaissances testés.
Nous avons également testé avec succès les deux approches de partitionnement que nous avons présentées en partie D.IV.3.4. Nous avons pu mettre en avant les avantages et les défauts de chacune de ces deux approches : l'approche basée sur l'apprentissage de nouvelles règles par apprentissage artificiel et sur la comparaison de ces règles avec les règles initiales (cf. D.IV.3.4.3) a donné en général de meilleurs résultats et a permis la génération de moins de partitions. Elle est toutefois limitée par sa complexité algorithmique. Elle devient ainsi inutilisable dès que le nombre de règles apprises devient trop élevé (de l'ordre de 15). L'approche basée sur la discrétisation indépendante de chacune des mesures (cf. D.IV.3.4.2), bien que moins efficace, n'a pas ces problèmes de complexité algorithmique.
Nous avons également testé les diverses approches d'exploration présentées en partie D.IV.3.5. Ces tests ont montré que le filtrage agent du voisinage permet, si le coefficient de filtrage est bien choisi, au processus de résolution de problèmes de converger plus rapidement vers la meilleure solution.
Nous avons enfin montré l'intérêt de la phase de simplification des bases de règles présenté en partie D.IV.3.6.

Deux aspects importants de notre approche de révision concernent le choix de l'échantillon utilisé comme support à la révision (échantillon de révision) et les jeux de mesures utilisés pour caractériser l'état de l'entité considérée.

Nous avons ainsi présenté en partie F.IV des expérimentations menées sur l'influence de l'échantillon de révision sur la qualité des connaissances pouvant être obtenues après révision. Ces expérimentations ont permis de vérifier que notre approche de sélection automatique d'échantillon de révision permet le choix d'un échantillon de révision pertinent.

Nous avons de même mené en partie F.V des expérimentations sur l'influence des jeux de mesures sur la qualité des connaissances obtenues par révision. Ces expérimentations ont permis de montrer l'influence des jeux de mesures sur les règles pouvant être obtenues après révision. Elles nous ont également permis de montrer la cohérence des résultats fournis par notre approche d'évaluation des jeux de mesures. Enfin, ces expérimentations nous ont permis de détecter les lacunes les plus importantes des jeux de mesures utilisés pour l'ensemble de nos expérimentations.

La dernière série d'expérimentations, présentée en partie F.VI, a permis d'évaluer notre approche de diagnostic de la qualité des jeux de connaissances. Nous avons ainsi pu vérifier que cette approche permettait de fournir un diagnostic en ligne cohérent des jeux de connaissances testés. Ces expérimentations ont également montré que les deux méthodes de prise de décision multicritère présentées en partie E.III.4 répondaient à notre problématique de diagnostic de la qualité des connaissances.

Nous avons enfin proposé dans une dernière partie (F.VII) une discussion sur les résultats de ces expérimentations. Nous avons ainsi évoqué l'intérêt de chercher à réviser les connaissances plutôt que de simplement chercher à en acquérir de nouvelles. Nous avons également soulevé les difficultés posées par la mise en œuvre de notre approche générale de révision pour un problème donné. La difficulté la plus importante concerne la définition de la fonction d'évaluation des performances qui a une importance cruciale dans notre approche. Nous avons à ce propos proposé quelques pistes pour répondre à ces difficultés.

Conclusion

Rappel de l'objectif de la thèse

Cette thèse s'inscrit dans le cadre de la résolution de problèmes par exploration informée d'arbres d'états. Ce type d'approche est largement utilisé pour la résolution de problèmes. Il nécessite néanmoins la définition de connaissances pertinentes pour guider le processus d'exploration. Malheureusement, la définition de ces connaissances, dites procédurales, s'avère souvent complexe.

L'objectif de ce travail de thèse était de proposer une approche de révision automatique des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états. Cet objectif était motivé par deux hypothèses. La première est que l'analyse d'instances résolues du problème considéré pourrait permettre d'améliorer les connaissances procédurales du système. La deuxième est qu'il est plus intéressant de chercher à réviser des connaissances déjà bonnes a priori plutôt que de chercher à en acquérir directement de nouvelles.

Cette thèse s'inscrit dans la continuité des travaux menés au laboratoire COGIT de l'Institut Géographique National sur la généralisation de données géographique. Nous avons donc posé comme objectif applicatif de mettre en œuvre cette approche de révision des connaissances dans le cadre d'un système dédié à la généralisation automatique de données géographique.

Bilan

Contributions au niveau de l'intelligence artificielle

Des méthodes pour la révision des connaissances par l'expérience

Notre approche générale de révision basée sur l'analyse de l'expérience nous a amenés à proposer diverses approches et méthodes qui constituent des apports au niveau de l'intelligence artificielle.

Un premier apport concerne la formalisation proposée du problème de la révision par l'expérience des connaissances procédurales contenues dans les systèmes fonctionnant par exploration informée d'arbres d'états.

Un autre apport est le modèle d'organisation des connaissances que nous avons proposé. Celui-ci est basé sur l'externalisation des connaissances procédurales du système de résolution de problèmes et sur leur modélisation sous la forme d'agents connaissance (B.V.2).

Concernant la phase de constitution de l'expérience de notre approche générale, nous pouvons évoquer deux apports principaux : la méthode de sélection automatique d'un échantillon représentatif d'instances du problème considéré (C.II.2) et le concept de jeu de connaissances *minimal* (C.III.1).

Concernant la phase de révision, un premier apport est notre approche de révision d'un ensemble de connaissances basée sur la décomposition en groupes de connaissances. Un autre apport, plus important, concerne l'approche proposée pour la révision des connaissances formalisées sous formes de bases de règles de production (D.III.3.3). Nous rappelons que celle-ci est basée sur trois étapes : la première consiste à partitionner l'espace des mesures,

utilisées pour définir les connaissances en zones admettant une conclusion a priori homogène. Nous avons proposé, dans ce cadre, deux méthodes de partitionnement (D.III.3.4). La seconde étape consiste à explorer l'espace des affectations de conclusions possibles afin de trouver l'affectation maximisant une certaine fonction d'évaluation (D.III.3.5). Nous avons proposé, pour cette seconde étape du processus de révision, une approche agents basée sur une analyse locale de la situation, pouvant être combinée à une recherche locale classique (D.III.3.5.5). Concernant la fonction d'évaluation, nous avons proposé un prototype de cette fonction. Ce prototype doit être spécialisé pour chaque application en fonction des besoins de l'utilisateur envers le système de résolution de problèmes (D.III.2.2). La dernière étape du processus de révision consiste à simplifier les bases de règles obtenues après révision (D.III.3.6).

Des méthodes de diagnostic

Nous avons proposé dans le cadre de cette thèse deux méthodes de diagnostic.
La première méthode de diagnostic proposée concerne l'évaluation des jeux de mesures utilisés pour la définition des connaissances (C.IV.3). Cette méthode se base sur la notion d'incohérence pour évaluer les jeux de mesures.
La seconde méthode de diagnostic proposée est dédiée au diagnostic en ligne de la qualité des connaissances (E.III). Cette méthode est basée sur l'analyse des instances résolues du problème considéré et sur l'utilisation d'une approche de décision multicritère.

Une application de l'intelligence artificielle

Un dernier apport concerne le caractère applicatif de ce travail de thèse. En effet, nous avons mis en œuvre notre approche dans le cadre d'un contexte applicatif réel, celui de la révision des connaissances d'un système de généralisation automatique de données géographiques. Cette mise en œuvre et les expérimentations menées ont montré en quoi les approches et les méthodes développées en intelligence artificielle pouvaient aider à la résolution de problèmes réels.

Contribution au niveau de la généralisation de données géographiques

Enrichissement du modèle de généralisation AGENT

Nous avons proposé d'enrichir le modèle de généralisation AGENT d'un point de vue formalisation et d'un point de vue cycle d'actions (B.V.3.1).

Nous avons ainsi proposé une formalisation pour les mesures utilisées par le modèle AGENT pour le calcul des valeurs courantes des contraintes et pour la définition des connaissances. Nous avons également proposé de formaliser le lien existant entre les contraintes et les actions.

Concernant le cycle d'actions, nous avons proposé d'enrichir le cycle d'actions du modèle AGENT par un test d'optimalité des états. Un état optimal désigne un état qu'il n'est pas possible d'améliorer compte tenu des actions disponibles.

Nous avons également proposé d'enrichir les connaissances relatives à la fin de cycle et à la validité des états. Nous avons, en effet, proposé de laisser la possibilité à l'utilisateur de définir la valeur de ces deux connaissances pour chaque classe d'agents géographiques.

Mise en œuvre de l'approche de révision des connaissances

Nous avons détaillé comment mettre en œuvre notre approche de révision des connaissances pour la révision des connaissances des systèmes de généralisation basés sur le modèle AGENT.
Nous avons proposé, dans ce cadre, des expérimentations pour la révision des connaissances relatives à la généralisation au 1 : 50 000 des *groupements de bâtiments* (cf. F).

Prototype implémenté

Ce travail de thèse nous a amenés à implémenter différents prototypes et outils qui constituent également un apport. Ces éléments ont été implémentés sur le système d'information géographique Clarity™.

- Implémentation d'un modèle dynamique de bases de règles de production. Dans Clarity™, ces bases de règles étaient écrites « en dur » dans le code. Nous avons proposé et implémenté un modèle afin que celles-ci puissent être modifiées dynamiquement pendant le processus de révision.
- Implémentation d'un prototype générique de notre modèle de révision. Nous avons implémenté l'ensemble des méthodes et des approches que nous avons proposées dans un modèle générique pouvant facilement être adapté à diverses utilisations.
- Réimplémentation du modèle AGENT. Nous avons implémenté le moteur du modèle AGENT afin qu'il intègre les enrichissements proposés (cf. B.V.3.1.1). Nous avons externaliser les connaissances sous la forme d'agents connaissance et intégré des *agents diagnostic*.
- Implémentation d'outils d'observation pour le modèle AGENT. Nous avons en particulier implémenté un outils de visualisation des arbres d'états obtenus après une généralisation et permettant de visualiser la géométrie des objets pour chaque état de l'arbre.

Enseignements

Notre travail est basé sur deux hypothèses : la première est que l'on peut améliorer la qualité des connaissances procédurales d'un système fonctionnant par exploration informée d'arbres d'états par analyse d'instances résolues du problème. La deuxième est qu'il est préférable pour de tels systèmes où les connaissances initiales sont généralement bonnes de procéder par révision des connaissances plutôt que de chercher à apprendre de nouvelles connaissances pour remplacer les connaissances initiales. Au vu des résultats obtenus pour les expérimentations menées sur la révision des connaissances du modèle de généralisation AGENT, nous considérons que ces deux hypothèses sont vérifiées (cf. F.III.8).

Ce travail a permis de soulever deux principales difficultés liées à la révision par l'expérience de connaissances contenues dans un système fonctionnant par exploration informée d'arbres d'états. La première concerne l'adéquation entre les besoins de l'utilisateur et la fonction utilisée pour évaluer la qualité d'un jeu de connaissances. La seconde concerne la pertinence de l'échantillon d'instances utilisé pour constituer l'expérience. Pouvoir assurer que la fonction d'évaluation traduit précisément les besoins de l'utilisateur et que l'échantillon est parfaitement représentatif de l'ensemble des instances que l'utilisateur souhaite traiter permet de garantir qu'un jeu de connaissances révisé sera de qualité égale ou supérieure au jeu de connaissances initial. Malheureusement, ces deux conditions sont rarement vraies en pratique. Nous avons néanmoins proposé une méthode pour le problème du choix de l'échantillon représentatif qui a donné de bons résultats lors des expérimentations (cf. F.IV). La fonction d'évaluation est dépendante du domaine d'application. Sa définition est un point clef de la

révision des connaissances. Il est donc très important que cette fonction reflète réellement les besoins de l'utilisateur.

Perspectives

Expérimentations supplémentaires sur la révision des connaissances du modèle de généralisation AGENT

Les expérimentations menées dans le cadre de cette thèse nous ont amenés à tester notre approche de révision pour les connaissances du modèle AGENT relatives à la généralisation au 1 : 50 000 des *groupements de bâtiments* (cf. F.II). Nous avons ainsi révisé ces connaissances à partir d'un échantillon *de groupements de bâtiments* sélectionné dans la ville d'Orthez. Puis, nous avons testé les connaissances obtenues sur les *groupements de bâtiments* de la ville de Salies-de-Béarn. Les tests menés sur les *groupements de bâtiments* de la ville de Salies-de-Béarn ont montré que tous les jeux de connaissances obtenus après révision permettaient d'obtenir un meilleur compromis efficience/efficacité que leurs jeux de connaissances initiaux respectifs.
Une expérimentation intéressante que nous n'avons pas eu le temps de mener serait de tester les jeux de connaissances révisés sur une nouvelle zone a priori très différente des zones utilisées pour la révision et pour les tests. Une telle expérimentation permettrait de vérifier que les jeux de connaissances révisés sont réellement pertinents pour l'ensemble des *groupements de bâtiments* et pas seulement pour ceux appartenant à des villes du Sud-Ouest de la France.

Nos tests de révision ont porté uniquement sur un type d'agents géographiques (et sur une seule échelle). Une évaluation plus complète de notre approche nécessiterait de la tester pour d'autres classes d'agents géographiques et d'autres échelles. Dans ce cadre, la mise en œuvre de notre approche pour les agents *tronçon de route* serait particulièrement intéressante. En effet, ces agents sont complexes à généraliser et la définition de leurs connaissances procédurales pose problèmes. De plus, la généralisation de ces agents fait intervenir une connaissance procédurale supplémentaire : celle relative à l'ordre d'activation des agents *tronçon de route*. En effet, un agent *tronçon route* peut se diviser en plusieurs sous-agents *tronçon routes* qui sont ensuite activés. Un sous-agent *tronçon routes* peut être activé plusieurs fois (généralement limité à 2 fois) et peut, à son tour, se rediviser en sous-agents *tronçon routes*. L'ordre d'activation des agents *tronçon de route* peut avoir une grande influence sur la qualité du résultat finale. Il est possible d'utiliser notre approche pour réviser cette connaissance indépendamment des autres en modélisant le problème du choix de l'agent à activer à chaque itération sous la forme d'un problème de recherche dans un arbre d'états où chaque branche correspond à l'activation d'un agent. Notons que l'application de notre approche nécessiterait, dans un premier temps, de recenser l'ensemble des critères (mesures) pouvant apporter des informations pour la définition de cet ordre d'activation.

Aide à la définition d'une fonction de performance

Nous avons évoqué, en partie F.VII.2, les difficultés liées à la mise en œuvre de notre approche pour un système de résolution de problèmes donné. La principale difficulté réside dans la définition d'une fonction de performance réellement représentative des besoins de l'utilisateur. En effet, notre approche est basée sur l'utilisation d'une fonction estimant les performances du système de résolution de problèmes pour un échantillon d'instances du problème considéré. Or, la définition d'une telle fonction peut s'avérer complexe. En effet, définir une telle fonction nécessite de pouvoir formaliser précisément les besoins liés à

l'utilisation de ce système. En partie F.VII.2, nous avons suggéré d'utiliser les techniques de l'apprentissage artificiel afin de faire face à cette difficulté.

Une première piste serait d'utiliser une approche inspirée de celle proposée par (Ruas & Holzapfel, 2003) pour la caractérisation des alignements de bâtiments. Ainsi, un échantillon de résultats serait proposé à un expert. Celui-ci donnerait pour chacun des résultats une note d'efficacité, une note d'efficience et une note pour les performances globales du système. Les résultats notés seraient ensuite utilisés pour apprendre, par apprentissage artificiel, une fonction d'évaluation de l'efficience, une fonction d'évaluation de l'efficacité ainsi qu'une fonction d'évaluation des performances globales dépendant des deux notes précédentes.

Une seconde solution, plus complexe, consisterait à définir cette fonction à l'aide d'un apprentissage actif. Il serait intéressant d'utiliser une approche inspirée de celle proposée par (Hubert, 2003 ; Christophe, 2008) pour la définition de légendes de cartes. Le système proposerait différents échantillon de résultats à l'utilisateur (typiquement les mêmes instances résolues avec des jeux de connaissances différents). L'utilisateur choisirait alors quels sont les meilleurs résultats et pourrait, par le biais d'une interface, ajouter des commentaires. Le système pourrait utiliser les commentaires de l'utilisateur pour affiner ses fonctions d'évaluation de l'efficacité, de l'efficience et des performances. Il pourrait également proposer d'autres échantillons pour affiner encore plus ses fonctions. L'apprentissage résulterait donc d'un dialogue entre l'utilisateur et le système.

Application à d'autres systèmes

Notre approche de révision des connaissances pourrait être adapté pour les systèmes basés sur le modèle de généralisation CartACom (Duchêne, 2004). Comme le modèle AGENT, le modèle CartACom est basé sur une modélisation agents des objets géographiques. Sa principale différence avec le modèle AGENT est qu'il propose une communication directe entre les agents, sans structure hiérarchique. Le choix des opérations de généralisation à appliquer résulte d'une concertation entre les agents impliqués dans un même conflit cartographique. Un point important dans ce modèle concerne l'ordre d'activation des agents. A chaque itération du processus de généralisation, l'agent activé est sélectionné par une heuristique. Nous pouvons remarquer que ce problème de choix de l'agent à activer à chaque itération est très proche du problème de l'ordre d'activation des agents *tronçon de route* dans le modèle AGENT. Une différence concerne la complexité du problème. Ici, le nombre d'agents concernés sera généralement beaucoup plus important. De plus, le nombre d'activations de chaque agent n'est pas limité. Il ne sera donc peut-être pas possible d'appliquer exactement la même méthode de révision que pour le problème de l'ordre d'activation des agents *tronçon de route* dans le modèle AGENT.

Une seconde application possible dans le domaine de la généralisation concerne le domaine de l'orchestration de systèmes de généralisation. Des travaux s'intéressent au problème de l'utilisation conjointe de plusieurs systèmes de généralisation (Touya, 2008). L'un des enjeux est alors de déterminer quel(s) système(s) utiliser pour une zone ou un thème donné. Si ce choix est évident pour certains thèmes (typiquement le modèle GAEL est spécialement dédié à la généralisation des thèmes champs), il est parfois plus complexe quand plusieurs systèmes sont susceptibles d'être utilisés pour une même zone et pour des thèmes communs. Obtenir une bonne généralisation pourra demander, pour certaines zones, d'appliquer plusieurs systèmes de généralisation. Le problème du choix des systèmes de généralisation à appliquer sur une zone peut être résolu à l'aide d'une stratégie d'exploration informée d'arbres d'états

où chaque branche de l'arbre correspond à l'application d'un système de généralisation. Notre approche de révision automatique pourrait être appliquée pour réviser les connaissances procédurales propres à ce problème de choix des systèmes de généralisation à appliquer.

Enfin, de part sa généricité, notre approche de révision peut facilement être adaptée à d'autres systèmes fonctionnant par exploration informée d'arbres d'états. L'utilisation d'une stratégie d'exploration informée d'arbres d'états étant une approche classique en résolution de problèmes, le nombre de domaines pour lesquels notre approche peut être utilisée, ou peut au moins servir de piste de réflexion, est donc très important.

Annexes

Annexe 1 : Algorithmes de traitement des groupements de bâtiments développés

An.1.1 Introduction

Nous avons mis au point différents algorithmes dédiés à la généralisation des *groupements de bâtiments*.

Le premier que nous présentons en annexe *An.1.2* est destiné à la détection des bâtiments se trouvant dans les angles droits formés par des *tronçons de route*. Ces bâtiments servent en effet souvent de marqueurs visuels, il donc intéressant de les conserver (et donc de les détecter) dans le cadre du processus de généralisation.

Le second algorithme que nous proposons (en annexe *An.1.3*) est un algorithme de suppression de bâtiments basé sur le contexte local des bâtiments. Ainsi les bâtiments superposés aux routes et ceux superposés à d'autres bâtiments sont supprimés.

An.1.2 Algorithme de détection des bâtiments dans les coins

Cet algorithme a pour objectif de détecter les bâtiments se trouvant dans les « coins ». Un coin désigne ici un virage dans une route ou entre deux routes dont l'angle est proche de 90°.

Le principe de l'algorithme est le suivant : nous cherchons dans un premier temps à découper les *tronçons de route* qui admettent un virage proche de 90°.
Nous construisons ensuite un triangle au niveau de chaque intersection de *tronçons de route* formant un angle proche de 90°. La surface du triangle dépend de l'angle formé par les *tronçons de route*.
Les bâtiments superposés à ces triangles seront considérés comme étant dans un « *coin* ».

La figure An.1 donne un exemple du fonctionnement de l'algorithme. Dans une première phase le tronçon de route rose a été divisé en 4 *tronçons de route*. Durant la seconde phase, 4 triangles « coin » ont été construits. Dans cet exemple, il y a donc 3 bâtiments se trouvant dans un coin.

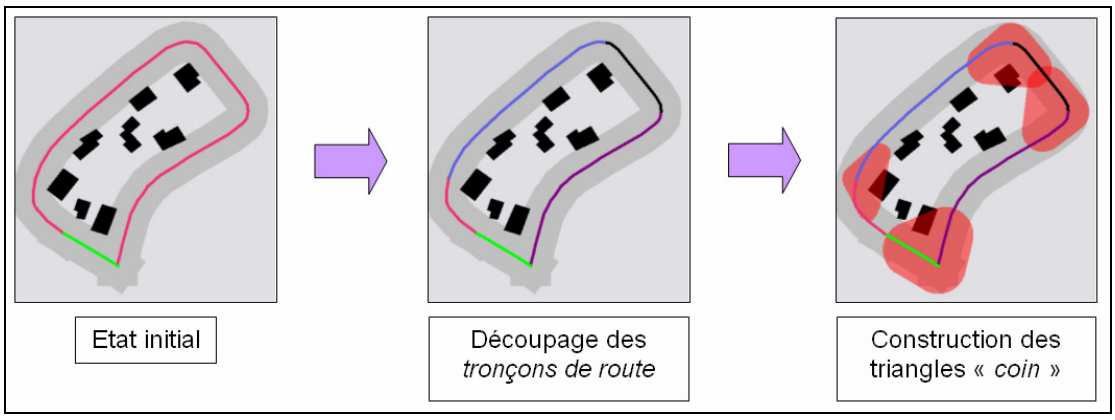


Figure An.1 Exemple de détection de bâtiments dans les « *coins* »

Nous définissons deux paramètres pour cet algorithme :

- TOLERANCE : tolérance acceptée pour les angles. Dans nos expérimentations TOLERANCE = 20°.
- COTE_TRIANGLE : taille des côtés des triangles représentant les coins. Dans nos expérimentation, COTE_TRIANGLE = 20 m.

```
Détection des bâtiments dans les coins(Agent groupement de bâtiments)
    //Découpage des tronçons de route qui admettent un virage à 90°
    Pour tous les tronçons de route tR du groupement de bâtiments faire :
        découpage(tRoute)
    Fin pour

    //Construction des surfaces « coin »
    Initialisation Ensemble de surfaces surfacesCoins ← {}//Ensemble
    contenant chaque surface représentant un coin
    Initialisation Ensemble de tronçons de route tronconTraites ← {}
    Pour tous les tronçons routiers tR1 du groupement de bâtiments faire :
        Ajouter tR1 à tronconTraites
        Pour tous les tronçons routiers tR2 du groupement de bâtiments
        connexe à tR1 faire :
            Si tronconTraites ne contient pas tR2 faire :
                Si [angle(tR1,tR2) > 90° - TOLERANCE] ET [angle(tR1,tR2) <
                90° + TOLERANCE] faire :
                    Initialisation Point A ← point connexe entre tR1 et
                    tR2
                    Initialisation Point B ← point se trouvant sur tR1 tel
                    que distance de AB :
                                AB = COTE_TRIANGLE × 90° / |90° − angle(tR1,tR2)|

                    Initialisation Point C ← point se trouvant sur tR2 tel
                    que distance de AC :
                                AC = COTE_TRIANGLE × 90° / |90° − angle(tR1,tR2)|

                    Initialisation Surface surfCoin ← triangle formé par
                    les points A, B et C
                    Ajouter surfCoin à surfacesCoins
                Fin si
            Fin Si
        Fin pour
    Fin pour
    //Détection des bâtiments dans les coins
    Initialisation Ensemble de bâtiments batimentsCoins ← {} //Ensemble des
    bâtiments se trouvant dans un coin
    Pour tous les bâtiments bat du groupement de bâtiments faire :
        Pour toutes les surfaces surfCoin dans surfacesCoins faire :
            Si bat et surfCoin sont superposés faire :
                Ajouter bat à batimentsCoins
            Fin si
        Fin pour
    Fin Tant que
    Retourner batimentsCoins
```

```
Découpage d'un tronçon de route(Agent tronçon de route)
     Initialisation Entier nbPointsIt ← 0
     Initialisation Point ptPrec ← (0,0)
     Initialisation Point ptPrecPrec ← (0,0)
     Initialisation Réel angleCumul ← 0

     Pour tous les points pt composant le tronçon de route faire :
          Si nbPointsIt > 3 faire :
             angleCumul ← angleCumul + angle(ptPrecPrec, ptPrec, pt)
             Si angleCumul > 90° - TOLERANCE faire :
                   Couper le tronçon de route au niveau du point ptPrec
                   angleCumul ← 0
                   nbPointsIt ← 0
             Fin si
          Fin si
          nbPointsIt ← nbPointsIt + 1
          ptPrecPrec ← ptPrec
          ptPrec ← pt
     Fin pour
```

An.1.3 Algorithme de suppression locale

Cet algorithme est destiné à supprimer les bâtiments se superposant. Son principe est de chercher à supprimer en priorité les bâtiments qui sont les plus superposés à d'autres bâtiments.

La figure An.2 présente un exemple de résultat obtenu avec cette action. Seul le bâtiment en superposition avec les trois autres bâtiments a été supprimé, résolvant ainsi les problèmes de superposition entre bâtiments.

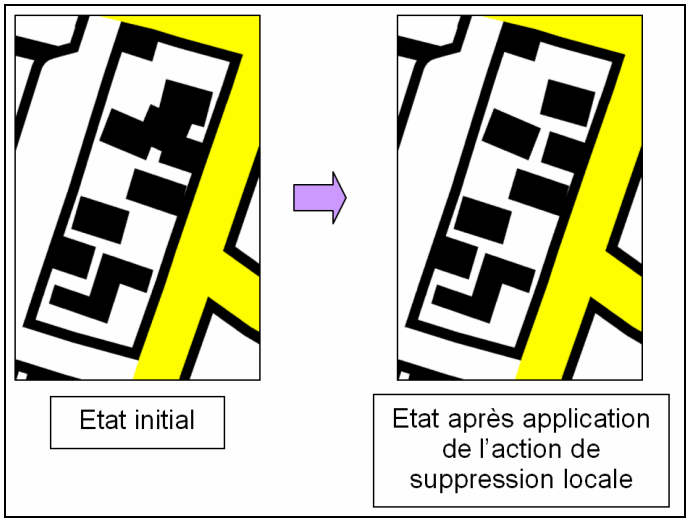


Figure An.2 Exemple d'application de l'action de suppression locale

```
Suppression locale de bâtiments (Agent groupement de bâtiments)
    //Instanciation du dictionnaire des superpositions
    Initialisation Dictionnaire dictSuperpositions //dictionnaire contenant
    pour chaque bâtiment l'ensemble des bâtiments avec lesquels il est
    superposé. Clef : bâtiment -> valeur : Ensemble de bâtiments.
    Entier nb_superpositionMax ← 0
    Pour tous les bâtiments bat1 du groupement de bâtiments faire :
        Pour tous les bâtiments bat2 du groupement de bâtiments faire :
            Si bat1 et bat2 sont différents faire :
                Initialisation Ensemble de bâtiments EnsSuppBat1 ← ensemble
                se trouvant à la clef bat1 du dictionnaire
                dictSuperpositions
                Si EnsSuppBat ne contient pas contient bat1
                    Si bat1 et bat2 sont superposées faire :
                        - Ajouter le bâtiment bat2 à EnsSuppBat1
                        - Ajouter le bâtiment bat1 à l'ensemble des
                        bâtiments EnsSuppBat2 se trouvant à la clef bat2
                        du dictionnaire dictSuperpositions
                        - nb_superpositionMax ← Max(nb_superpositionMax,
                        taille de EnsSuppBat1, taille de EnsSuppBat2)
                    Fin si
                Fin si
            Fin Si
        Fin pour
    Fin pour

    //Suppression des bâtiments superposés
    Tant que [nb_superpositionMax > 0] ET [nombre de bâtiments dans le
    groupement de bâtiments > 1] faire :
        Pour tous les bâtiments bat du groupement de bâtiments faire :
            Si l'ensemble des bâtiments EnsSuppBat se trouvant à la clef
            bat du dictionnaire dictSuperpositions à une taille égale à
            nb_superpositionMax faire :
                - Supprimer bat
                - Mettre à jour le dictionnaire dictSuperpositions pour
                  enlever dans tous les ensembles de bâtiments, le
                  bâtiment bat.
            Fin si
        Fin pour
    Fin Tant que
```

Il est possible d'ajouter à l'algorithme des contraintes de façon à ce que celui-ci ne supprime pas les grands bâtiments ou les bâtiments importants.

Annexe 2 : Présentation des algorithmes utilisés

An.2.1 Introduction

Nous présentons dans cette annexe les algorithmes d'apprentissage supervisé et non supervisé que nous avons utilisés dans nos expérimentations ainsi que ceux de sélection et de discrétisation supervisé d'attributs.

An.2.2 Algorithme C4.5

Introduction

L'algorithme C4.5, qui a été développé par Quinlan en 1993 (Quinlan, 1993), est une amélioration de son précédent algorithme, ID3 (Inductive Decision Tree) (Quinlan, 1986). Il fait partie de la famille des algorithmes d'induction d'arbres de décision. Cette famille d'algorithmes est très souvent utilisée en apprentissage. Les arbres de décision sont des classificateurs pour des instances représentées dans un formalisme attributs/valeurs. L'un des gros points forts de cette représentation en arbre de décision est sa lisibilité.

La classification d'un objet est réalisée par une suite de tests sur ses attributs. Ce sont les nœuds de l'arbre de décision qui testent les attributs. Il existe une branche pour chaque valeur de l'attribut testé. Les feuilles spécifient la classe à laquelle appartient l'objet.
La figure An.3 présente un exemple d'arbre de décision. Celui-ci a pour objectif de définir l'espèce d'un iris (parmi trois espèces : *iris-setosa*, *iris-versicolor* et *iris-virginica)* en fonction de la longueur et de la largeur de ses pétales.

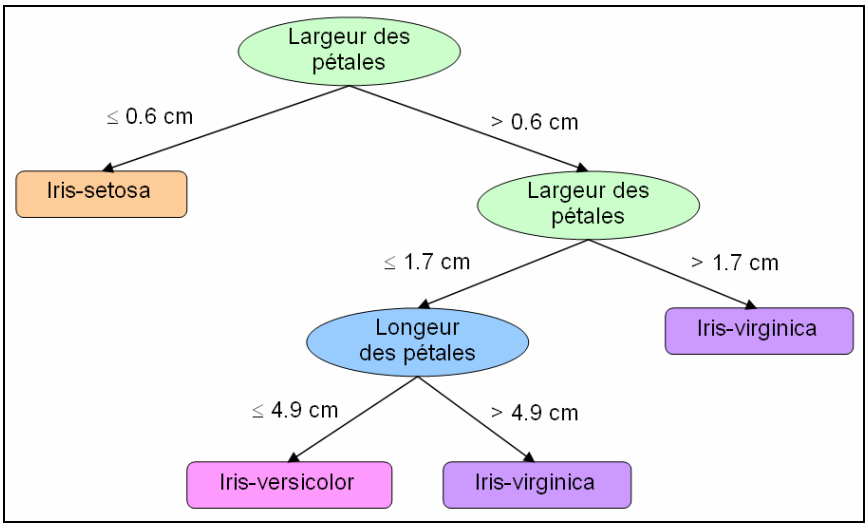


Figure An.3 Exemple d'arbre de décision

Induction d'un arbre de décision

L'objectif de l'induction d'un arbre de décision est de construire un arbre de décision à partir d'un jeu d'apprentissage composé d'exemples représentés dans un formalisme attributs/valeur. Un nombre important d'arbres de décision peut être construit pour un jeu d'apprentissage. Il est donc nécessaire de disposer d'une stratégie d'exploration des hypothèses efficace.

L'approche utilisée par C4.5 est l'approche TDIDT (*top-down induction of decision tree*). Elle procède de manière descendante en débutant par un arbre n'étant constitué que d'un nœud racine correspondant à une partition simple de l'espace des exemples. Elle construit l'arbre en choisissant un attribut à chaque étape de façon récursive jusqu'a atteindre le critère d'arrêt (plus d'attribut à tester ou obtention d'un échantillon pur, c'est-à-dire d'un échantillon dans lequel tous les exemples appartiennent à la même classe).

L'algorithme est le suivant :

```
ConstruireArbre (Ensemble d'exemples X)
  Si tous les exemples de X ont la même classe où s'il n'y a plus
  d'attribut à tester faire:
     Créer une feuille associée à la classe majoritaire
  Sinon faire :
     Choisir le meilleur couple (attribut/test) pour créer n nœud
     Séparer X en deux sous ensembles d'exemples Xd et Xg en fonction de
     ce couple
     ConstruireArbre(Xg)
     ConstruireArbre(Xd)
  Fin si
```

Pour résoudre la question du choix de l'attribut pour chaque étape, C4.5 se base sur le critère d'entropie. La notion d'entropie utilisée provient de Shannon qui a proposé en 1949 une mesure d'entropie valable pour les distributions discrètes de probabilité. Elle exprime la quantité d'information.

L'entropie est égale à :

$$I = -\sum_{i=1}^{k} p_i \times \log_2(p_i)$$

Avec p_i : probabilité de la classe C_i

On définit à partir de l'entropie, le gain d'entropie (figure An.3).

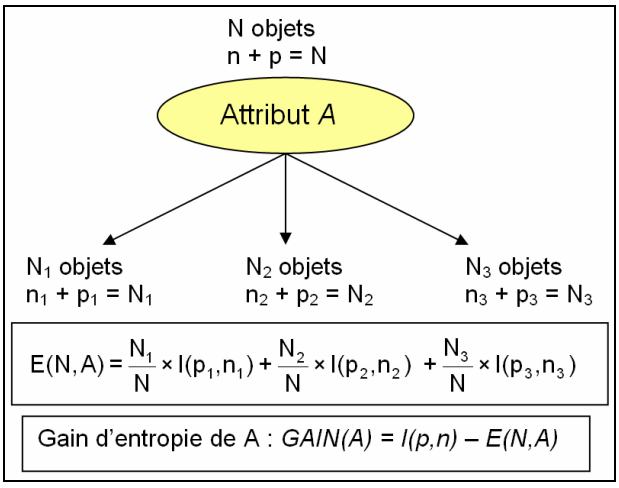


Figure An.3 Exemple d'application de l'action de suppression locale

Pour un cas à deux classes, la formule de l'entropie peut être simplifiée par :

$$I = P \times log_2 (P) - (1 - P) \times log_2 (1 - P)$$

C4.5, au contraire d'ID3, ne prend pas directement en compte ce gain d'entropie, mais le normalise par l'entropie de la variable prédictive. On appelle ce critère, le rapport de gain.

$$IGR(A) = \frac{GAIN(A)}{I}$$

Une autre amélioration apportée par C4.5 par rapport à ID3 concerne la gestion des attributs continus. Cette gestion se base sur le fait que l'ensemble d'exemples est fini, donc le nombre de valeurs des attributs sur ces exemples est aussi fini. Le principe de segmentation des variables continues utilisé est le suivant : on trie tous les points de coupure présents dans le jeu d'apprentissage selon la valeur de l'attribut, puis on cherche le seuil qui maximise le critère de segmentation.

Un problème posé est celui du surapprentissage. Que ce soit pour des raisons de présence de bruits dans les données d'apprentissage ou pour des raisons de complexité de la fonction cible à apprendre, l'arbre obtenu peut se révéler trop "précis" et trop complexe.
De façon à améliorer la généralisation ainsi que la compréhensibilité de l'arbre construit un élagage est réalisé. Cet élagage peut intervenir soit pendant la construction de l'arbre, on parle alors de pré-élagage, soit sur l'arbre obtenu après la construction, il s'agit alors d'un post-élagage.

Ce pré-élagage a pour rôle de modifier le critère de terminaison d'exploration d'une branche de l'algorithme de construction de l'arbre. Sans pré-élagage, ce critère est que l'on arrête d'explorer une branche dès qu'on a fini de tester tous les attributs ou dès que l'on obtient un échantillon pur (exemples appartenant tous à la même classe). C4.5, tout comme ID3, permet de fixer comme critère d'arrêt un seuil minimal de gain d'information.

L'une des stratégies utilisées par un certain nombre d'implémentations de C4.5 pour le post-élagage est celle du "*Reduced error prunning"* (Quinlan, 1987). Cet algorithme divise le jeu d'apprentissage en deux sous jeux (2/3 - 1/3). Le premier sert à la construction de l'arbre de décision, le second à son élagage. Pour chaque nœud *n* de l'arbre construit *A*, l'algorithme crée *A'* l'arbre *A* dans lequel on a remplacé *n* par une feuille. Si l'erreur sur le jeu d'exemples d'élagage obtenu avec l'arbre *A'* est inférieure à celle obtenue avec l'arbre *A* alors on garde *A'*. On répète cette opération jusqu'à ce que l'on n'ait plus rien à remplacer.

An.2.3 Algorithme EM

L'algorithme EM (Expectation Maximisation) a été proposé par (Dempster et al., 1977). Il est souvent utilisé en classification non supervisée pour diviser un ensemble d'objets en « cluster », c'est-à-dire en groupes d'objets. Son principe est de maximiser un critère de vraisemblance en se basant sur deux étapes : l'estimation et la maximisation.

Soit un ensemble d'exemples $S = \{x_1,\ldots,x_m\}$. Chaque exemple est décrit par un vecteur d'attributs. On fait l'hypothèse que ces vecteurs sont des tirages aléatoires d'un mélange de k distributions gaussiennes $N_1,\ldots N_k$.

On complète les exemples x_i par un vecteur z : $y_i = (x_i, z_{i1},\ldots,z_{ik})$ avec x_i étant le vecteur observé et pour tout j = 1,…,k, z_{ij} vaut 1 si x_i a été généré par N_i.
L'algorithme est le suivant :

```
EM(Ensemble d'exemples X)
   Initialiser aléatoirement h ←(μ1,…, μk)
   Tant que le processus n'a pas convergé faire :
        /*Estimation : calcul des estimations E(zi1) de zi1,…, E(zik) de zik,
           en supposant que l'hypothèse courante sur h est la bonne : */
        Pour j ← 1..k, faire :
                                   E(z_ij) ← p(x_i|μ_j) / Σ_{n=1}^{k} p(x_i|μ_n)
        Fin pour

        /*Maximisation : Calculer une nouvelle estimation de h ← (μ1,…,
           μk) en supposant que zij est égale à E(zij) :*/
        Pour j ← 1..k, faire :
                                   μ_j ← Σ_{i=1}^{k} E(z_ij) x_i / Σ_{n=1}^{k} E(z_ij)
        Fin pour
   Fin tant que
```

L'algorithme EM renvoie le vecteur $h = (\mu_1,...,\mu_k)$ et les estimations des valeurs $z_{i1},\ldots, z_{ik}$. Ces valeurs sont ensuite utilisées pour classer chaque exemple dans l'un des groupes.

An.2.4 Algorithme de sélection d'attributs

Nous présentons dans cette partie l'algorithme de sélection d'attributs proposé par (Hall & Smith, 1998).

Cet algorithme est basé sur l'exploration de l'espace des sous-ensembles d'attributs. Il recherche ainsi, parmi tous les sous-ensembles d'attributs possibles, celui qui maximise une certaine fonction d'évaluation. L'évaluation des sous-ensembles d'attributs est basée sur le coefficient de corrélation entre le sous-ensemble d'attributs et la classe ainsi que sur le coefficient d'intercorrélation du sous-ensemble.
Les sous-ensembles ayant un fort coefficient de corrélation avec la classe et un faible coefficient d'intercorrélation sont préférés.

La fonction d'évaluation d'un sous-ensemble d'attributs est la suivante :

$$G_s = \frac{k \times \overline{r_{ci}}}{\sqrt{k + k(k-1)\overline{r_{ii'}}}}$$

où k désigne le nombre d'attributs dans le sous-ensemble, $\overline{r_{ci}}$ désigne la corrélation moyenne avec la classe et $\overline{r_{ii'}}$ l'intercorrélation moyenne dans le sous-ensemble.

Différentes fonctions peuvent être utilisées pour le calcul des coefficients de corrélation et d'intercorrélation.

Nous avons utilisé pour le coefficient de corrélation le gain d'entropie (cf An.2.2), et pour l'intercorrélation le coefficient d'incertitude symétrique. Ce dernier est égal à :

$$G_s = 2 \times \frac{GAIN}{\sqrt{I(X) + I(Y)}}$$

Avec GAIN le gain d'entropie, et I(X) l'entropie de l'attribut X.

An.2.5 Algorithme de discrétisation d'attributs

Nous présentons dans cette partie l'algorithme de discrétisation proposé par (Fayyad & Irani, 1992). Cet algorithme est l'un des plus utilisés pour la discrétisation supervisée d'attributs numériques.

Le principe de cet algorithme est de diviser récursivement l'intervalle de valeur de l'attribut en 2 en utilisant le gain d'entropie pour décider du meilleur point de coupure.

L'algorithme est le suivant :

```
DiscretiseAttribut (Ensemble d'exemples X, intervalle [a,b])
  Si critère d'arrêt n'est pas atteint faire:
      Sélection de points de coupure
      Evaluation de chaque point de coupure par le calcul du gain
      d'entropie sur X
      Sélection du point de coupure Ptc qui maximise le gain d'entropie
      Séparer X en deux sous ensemble d'exemples Xa et Xb en fonction de ce
      point de coupure
      DiscretiseAttribut(Xa, ]a,Ptc])
      DiscretiseAttribut(Xb, ]Ptc,b[)
  Fin si
```

Les points de coupure désignent les valeurs de l'attribut pour lesquelles la classe change de valeur.
Par exemple, dans le cas d'exemples de la forme (Valeur attribut, classe) : (1, G), (1.5, G), (1.7, G), (2, H), (3, H), (7, G), les deux point de coupures sont 1.85 et 5. Ils marquent la valeur intermédiaire entre deux exemples de classes différentes.

(Fayyad & Irani, 1992) utilise le gain d'entropie (cf. An.2.2) pour sélectionner le point de coupure a priori le plus intéressant.

An.2.6 Recherche locale taboue

La recherche locale taboue (Glover, 1989), aussi appelé recherche avec tabous, fait partie de la famille des algorithmes de recherche locale. Son principe est de construire une première solution puis de chercher à améliorer cette solution en explorant son voisinage tout en mémorisant les derniers mouvements faits (la liste taboue) afin d'éviter les cycles.

L'algorithme est le suivant :

```
Recherche Locale Taboue()
  Initialisation d'une solution initiale e
  Initialisation Liste listeTaboue = {}
  Tant que critère de fin non atteint et qualité de e insuffisante, faire :
     Choisir dans le voisinage de e, la solution e' telle que :
        • Le mouvement de e->e' n'appartient pas à ListeTaboue
        • La fonction objectif soit maximale pour e'
     Remplacer e par e'
     Si longueur de ListeTaboue = longueurMax, faire :
           Enlever le plus vieux mouvement de ListeTaboue
     Fin si
     Ajouter le mouvement e->e' dans ListeTaboue
  Fin Tant que
```

La longueur de la liste longueurMax permet de privilégier soit la diversification (si longueurMax est faible) soit l'intensification (si longueurMax est élevé).
Dans le cas où cette longueur est nulle, on se retrouve avec une recherche gloutonne. Quand longueurMax est petit, la recherche a des risques de rester bloquée autour d'optima locaux. Plus longueurMax augmente, plus la recherche est diversifiée mais plus on risque de s'interdire de monter sur le pic optimal.

Des travaux ont proposé d'utiliser une liste de taille dynamique qui s'adapte au contexte (Battiti & Tecchiolli, 1994).

Glossaire

Agent : « entité logicielle ou physique à qui est attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome et en coopération avec d'autres agents » (Briot & Demazeau, 2001).

Agent connaissance : agent qui a pour tâche de gérer une connaissance. Il fournit des réponses aux requêtes du système de résolution de problèmes au sujet du contenu de la connaissance, il diagnostique la qualité et participe au processus de révision.

Agent diagnostic : agent qui a pour rôle de diagnostiquer la qualité globale des connaissances.

Agent géographique : objet géographique qui a pour but de se généraliser.

Apprentissage (artificiel) : notion qui « [...] englobe toute méthode permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle » (Cornuéjols et al., 2002, p.vi).

Apprentissage non supervisé : cas particulier d'apprentissage où l'apprenant doit trouver par lui-même, dans un ensemble d'observations, les régularités sous-jacentes.

Apprentissage supervisé : cas particulier d'apprentissage où l'apprenant cherche à apprendre, à partir d'un ensemble restreint d'observations étiquetées par un expert (aussi appelé oracle), une fonction (appelée hypothèse) permettant de reproduire ou d'approximer l'étiquetage de l'expert.

Analyse multicritère : méthode qui permet d'orienter un choix sur la base de plusieurs critères. Elle est essentiellement destinée à la résolution de problèmes de décision.

Arbre d'états : structure arborescente où chaque nœud correspond à un état de l'entité considérée et où l'application d'une action sur l'entité correspond à une branche. L'état initial de l'entité correspond à la racine de l'arbre.

Cohérence d'un jeu d'exemples : un jeu d'exemple est dit cohérent s'il n'existe pas deux exemples ayant des observations identiques et une classification différente.

Contrainte : « propriété imposée à un caractère » (Ruas, 1999).

Cycle de vie : succession d'actions appliquées par un agent lors de son activation.

Exemple (en apprentissage supervisé) : un exemple est composé d'une observation et d'une sortie désirée (appelée classe).

Généralisation (sens général et sens en IA) : « opération par laquelle, reconnaissant des caractères communs entre plusieurs objets singuliers, on réunit ceux-ci sous un concept unique dont ces caractères forment la compréhension » (Lalande, 1997).

Généralisation cartographique : cas particulier de généralisation dans laquelle les données dérivée sont destinées à être affichées de manière lisible à l'écran ou imprimées sur papier à une échelle donnée.

Généralisation de données géographiques : processus visant à diminuer le niveau de détail de données géographiques afin que celles-ci répondent à un besoin particulier (e.g. changement d'échelle).

Heuristique : stratégie de résolution de problèmes basée sur des connaissances spécifiques au problème considéré. Les heuristiques désignent aussi les connaissances empiriques utilisées par la méthode de résolution de problèmes.

Hypothèse (en apprentissage supervisé) : fonction permettant de passer de l'espace des observations à celui des classes.

Induction : passage d'observations particulières à une loi générale.

Instance d'un problème : Cas particulier de problème. Un exemple d'instance d'un problème consistant de trouver un plus court chemin entre deux villes peut être de trouver le plus court chemin entre Paris et Marseille.

Métaheuristique : stratégie de résolution de problèmes permettant de guider la recherche d'une solution optimale et qui peut être appliquée avec peu de modifications à différents problèmes.

Niveau meso : « Niveau des groupes d'objets géographiques » (Ruas, 1999).

Niveau micro : « niveau des objets géographiques » (Ruas, 1999).

Observations (en apprentissage supervisé) : partie descriptive d'un exemple.

Problème: « difficulté à résoudre pour obtenir un certain résultat » (Dictionnaire Petit Robert, 2007). Un exemple de *problème* peut être celui de trouver un plus court chemin entre deux villes.

Révision des connaissances : intégration de nouvelles informations dans une ou plusieurs bases de connaissances déjà définies.

Stratégie d'exploration informée : approche de résolution de problèmes s'appuyant sur des connaissances spécifiques au problème considéré pour le résoudre plus efficacement.

Surapprentissage (ou overfitting): cas où l'hypothèse apprise est trop spécifique à un échantillon d'exemples au détriment son pouvoirs de prédiction sur de nouveaux exemples. On parle aussi d'apprentissage par cœur.

Système multi-agent : « ensemble organisé d'agents » (Briot & Demazeau, 2001).

Index

A

B

C

E

F

G

I

J

M

N

P

R

S

T

V

Bibliographie

Aamondt, A. & Plaza, E. (1994), 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches', *AI Communication* **7(1)**, pp. 39--59.

Alchourron, C.; Gardenfors, P. & Makinson, D. (1985), 'On the Logic of Theory Change: Partial Meet Contraction and Revision Functions', *The Journal of Symbolic Logic* **50(2)**, pp. 510--530.

Angluin, D. (1988), Queries and concept learning, *Machine learning journal* **2**, pp.319--342.

Appriou, A. (1991), 'Probabilité et incertitude en fusion de données multi-senseurs', *Revue Scientifique et Technique de la Défense* **1**, pp. 27--40.

Atzmueller, M.; Puppe, F. & Baumeister, J. (2006), Introspective Subgroup Analysis for Interactive Knowledge Refinement, *in* 'Proceedings of the 19th Intl. Florida Artificial Intelligence Research Society Conference', Florida, USA, pp. 402--407.

Bader, M. (2001), 'Energy minimization methods for feature displacement in map generalisation', Thèse de l'université de Zurich.

Bader, M.; Barrault, M. & Weibel, R. (2005), 'Building displacement over a ductile truss', *international journal of geographical information science* **19**, pp. 915--936.

Baeck, T.; Fogel, D. & Michalewicz, Z., ed. (2003), *Evolutionary Computation 1, Basic Algorithm and Operators*, Institute of Physics (IOP).

Baeijs, C. (1998), 'Fonctionnalité émergente dans une société d'agents autonomes; étude des aspects organisationnels dans les systèmes multi-agents réactifs', Thèse de l'institut national polytechnique de Grenoble.

Bana e Costa, C. & Vansnick, J. C. (1999), The MACBETH approach: basic ideas, software and an application. *Advances in decision analysis*, pp. 131--157.

Bard, S. (2004), 'Méthode d'évaluation de la qualité de données géographiques généralisées. Application aux données urbaines', Thèse de l'université Paris VI, laboratoire COGIT. Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/BARD/These_Bard_2004.pdf

Barillot, X. (2002a), Characterization of complex bends, *in* 'Proceedings of the 10th GISRUK', Session 4C, pp. 179--181.

Barillot, X. (2002b), Mesures et structures d'analyse, 'Généralisation et représentation multiple', dir. Anne Ruas, Hermès Lavoisier, chap. 10, pp. 187--202.

Barrault, M.; Regnauld, N.; Duchêne, C.; Haire, K.; Baeijs, C.; Demazeau, Y.; Hardy, P.; Mackaness, W.; Ruas, A. & Weibel, R. (2001), Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization, *in* '20th international conference of cartography', pp. 2110--2116.

Battiti, R. & Tecchiolli, G. (1994), 'The reactive tabu search', *ORSA Journal on Computing* **6(2)**, pp. 126--140.

Beard, K. (1991), Constraints on rule formation, *in* Barbara Buttenfield & Robert McMaster, ed.,'Map generalisation: making rules for knowledge representation', Longman Scientific and Technical, Harlow, Essex, pp. 121--135.

Bellman, R. (1957), 'A Markov decision process', *Journal of Mathematical Mech.* **6**, pp. 679--684.

Benayoun, R.; Laritchev, O.; de Mongolfier, J. & Tegny, J. (1971), 'Linear programming with multiple objective functions: STEP method (STEM)', *Math. Program.* **1(3)**, pp. 366--375.

Ben Mena, S. (2000), 'Introduction aux méthiodes multicritères d'aide à la décision', *Biotechnol. Agro. Soc. Environ.* **4(2)**, pp. 83--93.

Beynon, M.; Curry, B. & Morgan, P. (2000), 'The Dempster-Shafer theory of evidence : An alternative approach to multicriteria decision modelling', *Omega, International Journal of Management Science* **28**, pp. 37--50.

Birattari, M.; Paquete, L.; T., S. & K., V. (2001), 'Classification of Metaheuristics and Design of Experiments for the Analysis of Components', Technical report, Darmstadt, University of Technology.

Bishop,C.M.(1995), Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK.

Blum, C. & Roli, A. (2003), 'Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison', *ACM Computing Surveys* **35**, pp. 268--308.

Blumer, A.; Ehrenfeucht, D.; Haussler, D. & M., W. (1989), 'Learnability and the Vapnik-Chevonenskis dimension', *Journal of the ACM* **36**, pp. 929--965.

Boffet, A. (2001), 'Méthode de création d'informations multi-niveaux pour la généralisation cartographique de l'urbain', Thèse de l'université de Marne la Vallée, Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/BOFFET/These_Boffet.zip

Boswell, R. & Craw, S. (1999), Organizing knowledge refinement operators, *in* 'Proceedings of the 5th European Symposium on Verification of Knowledge Based Systems and Components', Oslo, Norway, pp. 149--161.

Boullé, M. (2006), 'MODL : A Bayes optimal discretization method for continuous attributes', *Machine learning* **65(1)**, pp. 131--165.

Brans, J. & Vincke, P. (1985), 'A preference ranking organization method.', *Manage. Sci.* **31(6)**, pp. 647--656.

Brassel, K. & Weibel, R. (1988), 'A review and conceptual framework of automated map generalisation', *International Journal of Geographical Information Systems* **2**(3), pp. 229--244.

Brauers, W. (2007), 'What is meant by normalisation in decision making', *International Journal of Management and Decision Making* **8(5)**, pp.445--460.

Breiman, L.; Friedman, J. H.; Olshen, R. A. & Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth International.

Breslow L., Aha D. (1997), Simplifying Decision Trees: A survey, *The Knowledge Engineering Review* **12(1)**, pp.1--40.

Briot, J. & Demazeau, Y. (2001), *Principes et architecture des systèmes multi-agents*. Lavoisier.

Buchanan, B.; Sutherland, G. & Feigenbaum, A. (1969), 'Heuristic DENDRAL: a program for generating explanatory hypothesis in organic chemistry', *Machine Intelligence* **4**, pp. 209--254.

Bullnheimer, B.; Hartl, R. & C., S. (1997), Applying the Ant System to the Vehicle Routing Problem, *in* '2nd Metaheuristic International Conference', pp. 285--296.

Burghardt, D. & Meier, S. (1997), Cartographic displacement using the snakes concept, *in* W. Foerstner & L. Pluemer, ed.,'Semantic modelling for the acquisition of topographic information from images and maps', Birkhaeuser verlag, Basel, pp. 59--71.

Burghardt, D. & Neun, M. (2006), Automated sequencing of generalisation services based on collaborative filtering, *in* '4th International Conference GIScience', pp. 41--46.

Burghardt, D.; Neun, M. & Weibel, R. (2005), 'Generalization services on the web - classification and an initial prototype implementation', *Cartography and geographic information science* **32**(4), p. 257--268.

Carbonara, L. & Sleeman, D. (1999), 'Effective and Efficient Knowledge Base Refinement', *Machine Learning* **37**, pp. 143--181.

Caruana, R. & Freitag, D. (1994), Greedy Attribute Selection, *in* 'International Conference on Machine Learning', New Brunswick, New Jersey, Morgan Kauffmann, pp. 28--36.

Castro, A. & Oliveira, E. (2007), A Distributed Muti-agent system to solve airline operations problems, *in* 'ICEIS', Funchal, Madeira.

Cerny, V. (1985), 'Thermodynamical Approach to the Traveling Salesman Problem: an E_cient Simulation Algorithm.', *Journal of Optimization Theory and Applications* **45 (1)**, pp. 41--51.

Chaib-draa, B., Jarras, I. & Moulin, B. (2001), Systèmes multiagents: Principes généraux et applications. In Briot, J. P. and Demazeau, Y., editors, *Principes et architecture des systèmes multi-agents*. Hermes, Paris, France.

Cohen, W. (1995), Fast effective rule induction, *in* Proc. *International Conference on Machine Learning (ICML-95)*, pp. 115--123.

Collette, Y. & Siarry, P. (2003), *Multiobjective Optimization: Principles and Case Studies*, Interfaces.

Cornuéjols, A.; Miclet, L. & Kodratoff, Y. (2002), *Apprentissage artificiel : Concepts et algorithmes*, Eyrolles.

Christophe, S. (2008), Creative Cartography based on Dialogue, *in* 'In proceedings of AutoCarto', Shepherdstown, West Virginia, CD-ROM.

Davies, D. L. & Bouldin, D. W. (1979), A cluster separation measure, *in* 'IEEE Transactions on Pattern Analysis and Machine Intelligence', pp. 224--227.

De Loor, P.; Septseault, C. & Chevaillier, P. (2003), Les émotions : une métaphore pour la résolution de problèmes dynamiquement distribués, *in* 'JFSMA', Revue des sciences et technologies de l'information. Hammamet (Tunisie). n° hors série, pp. 331--344.

Dempster, A. (1967), 'Upper and lower probabilities induced by multivalued mapping', *Annals of Mathematical Statistics* **38**, pp. 325--339.

Dempster, A. P.; Laird, N. M. & Rubin, D. B. (1977), 'Maximum likelihood from incomplete data via the em algorithm (with discussion)', *Journal of the Royal Statistical Society* **39**, pp. 1--38.

Dorigo, M. & Stützle, T. (2004), *Ant Colony Optimization*, MIT Press.

Dougherty, J.; Kohavi, R. & Sahami, M. (1995), Supervised and Unsupervised Discretization of Continuous Features, *in* 'Proceedings of the 12th International Conference on Machine Learning', pp. 194--202.

Drogoul, A. (1993), 'De la simulation multi-agent à la résolution collectivede problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents.', Thèse de l'université Paris 6.

Drogoul, A.; Ferber, J. & Jacopin, E. (1991), Pengi : Applying Eco-Problem-Solving for behaviour Modelling in an Abstract Eco-System, *in* 'Proceedings of ESM, Simulation Councils, Copenhague, pp. 337--342.

Duchêne, C. (2004), 'Généralisation cartographique par agents communicants: le modèle CartACom', Thèse de l'université Pierre et Marie Curie Paris VI, laboratoire COGIT. Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/DUCHENE/These_Duchene_2004.pdf

Duchêne, C.; Dadou, D. & Ruas, A. (2005), Helping the capture of expert knowledge to support generalisation, *in* 'ICA Workshop on generalization and multiple representation'. La Corona, Spain. Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/Acoruna/Papers/Duchene_et_al.pdf

Duda, R., & Hart, P. (1973), Pattern Classification and Scene Analysis. Wiley-Interscience.

Eastman, J. (1985), 'Cognitive Models and Cartographic Design Research', *Cartographic Journal* **22(2)**, pp. 95--101.

Fayyad, U. & Irani, K. (1992), 'On the Handling of Continuous-Valued Attributed in Decision Tree Generation', *Machine Learning* **8**, pp. 87--102.

Feigenbaum, E. (1977), 'The art of artificial intelligence 1: Themes and case studies of knowledge engineering.', Technical report, Stanford University, Department of Computer Science.

Feo, T. & Resende, M. (1992), 'Greedy randomized adaptive search procedure', *Journal of Global Optimization* **42**, pp. 32--37.

Ferber, J. (1997), *Les systèmes multi-agents : Vers une intelligence collective*, InterEditions.

Figueira, J.; Mousseau, V. & Roy, B. (2005), ELECTRE Methods. *In*: Figueira, J., Greco, S., and Ehrgott, M., (Eds.), Multiple Criteria Decision Analysis: State of the Art Surveys, Springer, New York, pp. 133--162.

Fisher, D. (1987), 'Knowledge acquisition via incremental conceptual clustering', *Machine Learning* **2(2)**, pp. 139--172.

Fogel, L.; Owens, A. & Walsh, M. (1966), *Artificial Intelligence through Simulated Evolution*, Wiley.

Foerster, T. & Stöter, J. (2007), Towards A Formal Classification Of Generalization Operators, *in* 'International Cartographic Conference', Moscow, Russia, CD-ROM, Theme 10 : Cartographic generalization and multiple representation, Session Oral 8: Semantics and terminology in generalization, 16 pages.

Foerster, T. & Stöter, J. (2006), Establishing an OGC Web Processing Service for generalization processes, *in* 'workshop on generalisation and multiple representation', Portland, United-states, Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/Portland/paper/ICA2006-foerster_stoter.pdf

Fritsch, E. (1997), 'Représentations de la géométrie et des contraintes cartographiques pour la généralisation du linéaire routier', Thèse de l'université de Marne la Vallée, laboratoire COGIT. Disponible sur le site web de l'IGN : http://recherche.ign.fr/labos/cogit/pdf/THESES/FRITSCH/These_Fritsch_1997.zip

Gaffuri, J. (2008), 'Généralisation automatique à base d'opérations discrètes et continues pour la prise en compte des thèmes champ : le modèle GAEL', Thèse de l'université Paris Est. Disponible sur le site web de l'IGN : http://recherche.ign.fr/labos/cogit/pdf/THESES/GAFFURI/these_gaffuri.pdf

Gaffuri, J. & Trévisan, J. (2004), Role of urban patterns for building generalisation: An application of AGENT, *in* 'Workshop on Generalisation and Multiple representation', Leicester, United Kingdom. Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/Leicester/paper/Gaffuri-v2-ICAWorkshop.pdf

Gaines, B.R. & Compton, P.(1995), Induction of Ripple-Down Rules Applied to Modeling Large Databases. *Journal of Intelligent Information Systems* **5(3)**, pp. 211--228.

Galanda, M. & Weibel, R. (2002), An agent-based framework for polygonal subdivision generalization, *in* D. Richardson & P. van Oosterom, ed.,'Advances in spatial data handling, 10th international symposium on spatial data handling', Springer, Heidelberg, Berlin, Germany, pp. 121--136.

Gambardella, L.; Taillard, E. & Dorigo, M. (1999), 'Ant colonies for the quadratic assignment problem', *Journal of the Operational Research Society* **50**, pp. 167-- 176.

Gardenfors, P. (1985), *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, MIT Press.

Georgopoulou, E.; Sarafidis, Y.; Mirasgedis, S.; Zaimi, S. & Lalas, D. (2003), 'A multiple criteria decision-aid approach in defining national priorities for greenhouse gases emissions reduction in the energy sector', *European Journal of Operational Research* **146(1)**, pp. 199--215.

Ginsberg, A.; Weiss, S. M. & Politakis, P. (1988), 'Automatic knowledge base refinement for classification systems', *Artificial Intelligence* **35**, pp. 197--226.

Glover, F. (1986), Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research* **5**, pp. 340--349.

Gokgoz, T. (2005), 'Generalization of contours using deviation angles and error bands', *The Cartographic Journal* **42(2)**, pp. 145--156.

Goldberg, D. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Kluwer Academic Publishers.

Guérif, S. & Bennani, Y. (2007), Sélection de Variables en Apprentissage Numérique Non Supervisé, *in* 'Conférence francophone sur l'Apprentissage automatique', Grenoble, France, pp. 221--236.

Guyon, E. & A., E. (2003), 'An Introduction to Variable and Feature Selection', *Journal of Machine Learning Research* **3**, pp. 1157--1182.

Hagemana, J.; Wehrens, R.; van Sprang, H. & Buydens, L. (2003), 'Hybrid genetic algorithm–tabu search approach for optimising multilayer optical coatings', *Analytica Chimica Acta* **490**, pp. 211--222.

Hall, M. & Holmes, G. (2003), Benchmarking Attribute Selection Techniques for Discrete Class Data Mining, *in* 'IEEE Transactions on Knowledge and Data Engineering', vol. 15, pp. 1437--1447.

Hall, M. A. & Smith, L.A. (1998), Practical feature subset selection for machine learning, *in* 'Australian Computer Science Conference', Springer, pp. 181--191.

Hansen, P & Mladenović, N. (1997), Variable neighbourhood search for the *p*-median, *Location Science* **5**, pp. 207--226.

Harrie, L. (2001), 'An optimisation approach to cartographic generalisation', PhD thesis, Lund university, Lund institute of technology, department of technology and society.

Harrie, L. & Sarjakoski, T. (2002), 'Simultaneous graphic generalisation of vector data sets', *GeoInformatica* **6**(3), pp. 233--261.

Højholt, P. (2000), 'Solving space conflicts in map generalisation: using a finite element method', *Cartography and Geographic Information Sciences* **27**(1), pp. 65--74.

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press.

Hubert, F. (2003), 'Modèle de traduction des besoins d'un utilisateur pour la dérivation de données géographiques et leur symbolisation par le Web', Thèse de l'université de Caen, Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/HUBERT/These_Hubert_2002.pdf

Hubert, F. & Ruas, A. (2003), A method based on samples to capture user needs for generalisation, *in* 'fifth workshop on progress in automated map generalisation', Paris. Disponible sur le site web de la commission en généralisation de l'ACI :
http://aci.ign.fr/BDpubli/paris2003/papers/hubert_et_al_v0.pdf

Ignizio, J. (1978), 'A review of goal programming: a tool for multi objective analysis', *Journal of the Operational Research Society* **29**, pp.1109--1119.

Jabeur, N. (2006), 'A multi-agent system for on-the-fly web map generation and spatial conflict resolution', Thèse de l'université Laval.

Jacquet-Lagreze, E. & Siskos, J. (1982), 'Assessing a set of additive utility functions for multicriteria decision making, the UTA method', *European Journal of Operational Research* **10(2)**, pp. 151--164.

Jiang, L. & Zhang, H. (2006), Weightily Averaged One-Dependence Estimators, *in* 'Proceedings of the 9th Biennial Pacific Rim International Conference on Artificial Intelligence', pp. 970--974.

Jones, C. (1997), 'Geographical information systems and computer cartography', Addison Wesley Longman, pp. 271--289.

John, G. & Langley, P. (1995), Estimating Continuous Distributions in Bayesian Classifiers, *in* 'Eleventh Conference on Uncertainty in Artificial Intelligence', Montreal, Quebec, pp. 338--345.

Kelbassa, H. (2003), Selection of Optimal Rule Refinements, *in* 'Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference', pp.218--222.

Kilpeläinen, T. (2000), 'Knowledge Acquisition for Generalization Rules', *Cartography and Geographic Information Science* **27(1)**, pp. 41--50.

Kirkpatrick, S.; Gellatt, C. & M.P., V. (1983), 'Optimization by Simulated Annealing', *Science* **220**, pp.671--680.

Konieczny, S. (1999), 'Sur la logique du changement : révision et fusion de bases de connaissance', Thèse de l'université des sciences et technologies de Lille.

Laird, J.; Rosenbloom, P. & Newell, A. (1986), 'Chunking in soar: the anatomy of a general learning mechanism', *Machine Learning* **1**, pp. 11--46.

Lalande, A. (1997), *Vocabulaire technique et critique de la philosophie*, Presses Universitaires de France.

Lamy, S.; Ruas, A.; Demazeau, Y.; Baeijs, C.; Jackson, M.; Mackaness, W. & Weibel, R. (1999), AGENT Project: Automated Generalisation New Technology, in '*Proceedings of the 5th EC-GIS Workshop*' *(http://ams.egeo.sai.jrc.it/5ec-gis)*, Stresa, Italy.

Lee, D. (2003), Recent generalisation development and road ahead, *in* 'fifth workshop on progress in automated map generalisation', Paris. Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/BDpubli/paris2003/papers/lee_v0.pdf

Le Hégarat-Mascle, S.; Richard, D. & Ottlé, C. (2003), 'Multi-scale data fusion using Dempster-Shafer evidence theory', *Integrated Computer-Aided Engineering* **10(1)**, pp. 9--22.

Lemarié, C. (2003), Generalisation process for top100: research in generalisation brought to fruition, *in* 'fifth workshop on progress in automated map generalisation', Paris. Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/BDpubli/paris2003/papers/lemarie_v0.pdf

Li, Z. L. (2007), Essential Operations And Algorithms For Geometric Transformations In Digital Map Generalization, *in* 'International Cartographic Conference', Moscow, Russia, CD-ROM, Theme 10 : Cartographic generalization and multiple representation, Session Oral 8: Semantics and terminology in generalization, 17 pages.

Linden, J. & Stijnen, H. (1995), *QUALIFLEX version 2.3. A software package for multicriteria analysis*, Kluwer.

Lourenço, R. & Costa, J. (2004), 'Using ELECTRE TRI outranking method to sort MOMILP nondominated solutions', *European Journal of Operational Research* **153(2)**, pp. 271--289.

Ma, Y. & Wilkins, D. (1991), Improving the performance of inconsistent knowledge bases via combined optimization method, *in* 'Proceedings of the Eighth International Machine Learning Workshop', pp. 23--27.

Mackaness, W. & Ruas, A. (2007), Evaluation in the Map Generalisation Process, *in*: WA Mackaness, A Ruas, and LT Sarjakowski (eds), Generalisation of Geographic Information: Cartographic Modelling and Applications. Oxford, UK: Elsevier, pp. 89--112.

Mackaness, W. A. & Steven, M. (2006), 'An Algorithm for Localised Contour Removal over Steep Terrain', *The Cartographic Journal* **43**(2), pp. 144--156.

MacQueen, J.B. (1967), Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, pp. 281--297.

Marques Pereira, R. & Ribeiro, R. (2003), 'Aggregation with generalized mixture operators using weighting functions', *Fuzzy sets and systems* **137**, pp. 43--58.

Masud, A. & Hwang, C. (1981), 'Interactive sequential goal programming', *Journal of the Operational Research Society* **32(5)**, pp. 391--400.

McMaster, R. B. & Shea, K. S. (1988), Cartographic Generalization in a Digital Environment: a Framework for implementation in a GIS, *in* 'GIS/LIS'88', pp. 240--249.

McMaster, R. B. & Shea, K. S. (1992), *Generalization in digital cartography*, edited by the Association of American geographers, ISBN 8-89291-209-X.

Metaheuristics Network Website (2000), 'http://www.metaheuristics.net', visité en juillet 2008.

Milligan, G. & Cooper, M. (1985), 'An examination of procedures for determining the number of clusters in a data set', *Psychometrika* **50(2)**, pp. 159--179.

Minton, S. (1990), 'Quantitative results concerning the utility of explanation-based learning', *Artificial Intelligence* **42**, pp. 363--392.

Mitchell, T.; Keller, R. & Kedar-Cabelli, S. (1986), 'Explanation-based generalization: a unifying view', *Machine Learning* **1**, pp. 45--80.

Mitchell, T. (1980), 'The Need for Biases in Learning Generalizations', Technical report, Rutgers Computer Science Departement.

Mitra, P.; Murthy, C. & Pal, S. (2002), Unsupervised Feature Selection Using Feature Similarity, *in* 'IEEE Transactions on Pattern Analysis and Machine Intelligence', vol. 24, n°3, pp. 301--312.

Mladenovic, N. & Hansen, P. (1997), 'Variable Neighborhood Search', *Computers and Operations Research* **24(11)**, pp. 1097--1100.

Muggleton, S. (1995), 'Inverse entailment and progol', *New generatiob computing* **13(3-4)**, pp. 243--286.

Müller, J. & Wang, Z. (1992), 'Area-patch generalisation: a competitive approach', *The Cartographic Journal* **29(2)**, p. 137--144.

Mustière, S. (2005), 'Cartographic generalization of roads in a local and adaptative approach: a knowledge acquisition problem', *international journal of geographical information science* **19**(8--9), pp. 937--955.

Mustière, S. (2001), 'Apprentissage supervisé pour la généralisation cartographique', Thèse de l'université Pierre et Marie Curie Paris VI, laboratoire COGIT. Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/MUSTIERE/These_Mustiere_2001.pdf

Mustière, S. (1998), Généralisation adaptative du linéaire basée sur la détection d'empâtement, application au routier, *in* 'Bulletin d'information scientifique et technique de l'IGN n° 67, Recherche 97, IGN, pp. 33--42.

Mustière, S. & Lecordix, F. (2002), La généralisation du linéaire routier : des algorithmes et leur enchaînement, Généralisation et représentation multiple, dir. Anne Ruas, Hermès Lavoisier, chap. 13, pp. 241--256.

Mustière, S. & Ruas, A. (2004), Vers une réconciliation des experts et des systèmes, *in* 'Cassini 2004, 7ième conférence du GDR SIGMA', Grenoble, France.

Neri, F. & Saitta, L. (1994), Knowledge Representation in Machine Learning, *in* 'Proceedings of ECML'1994', Catania, Italy, pp. 20--27.

Neun, M. & Burghardt, D. (2005), Web Services for an Open Generalisation Research Platform, *in* 'workshop on generalisation and multiple representation', La Corona, Spain. Disponible sur le site web de la commission en généralisation de l'ACI :
http://aci.ign.fr/Acoruna/Papers/Neun_Burghardt.pdf

Newell, A. & Simon, H. (1956), 'The logic theory machine', *IRE Transactions on Information Theory* **2(3)**, pp. 61--79.

Olteanu-Raimond, A. & Mustière, S. (2008), Data matching-a matter of belief, *in* 'The International Symposium on Spatial Data Handling' (SDH), Montpellier, France, pp. 501--519.

Omrani, H.; Ion-Boussier, L. & Trigano, P. (2007), A new approach for impacts assessment of urban mobility, *in* 'WSEAS transaction on Information science and applications' **4(3)**, pp. 439--444.

Ourston, D. & Mooney, R. (1990), Changing the rules: A comprehensive approach to theory refinement, *in* 'Proceedings of the Eighth National Conference on Artificial Intelligence', pp. 815--820.

Patterson, D. (1996), *Artificial Neural Networks*, Singapore: Prentice Hall.

Pelleg, D. & Moore, A. (2000), X-means: Extending K-means with Efficient Estimation of the Number of Clusters, *in* 'Seventeenth International Conference on Machine Learning', pp. 727--734.

Platt, J. (1998), 'Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines', Technical report, Microsoft Research.

Plazanet, C. (1996), 'Enrichissement des bases de données géographiques: analyse de la géométrie des objets linéaires pour la généralisation cartographique (application aux routes)', Thèse de l'université de Marne La Vallée, Laboratoire COGIT. Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/PLAZANET/These_Plazanet_1996.zip

Plazanet, C.; Martini Bigolin, N. & Ruas, A. (1998), 'Experiments with Learning Techniques for Spatial Model Enrichment and Line Generalization', *GeoInformatica* **2(4)**, pp. 315--333.

Projet AGENT (2001), 'ESPRIT/LTR/24939', site internet.

Quinlan, J. (1993), *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc.

Quinlan, J. (1987), 'Simplifying Decision trees', *International Journal of Man-Machine Studies* **27**, pp. 221--234.

Quinlan, J. (1986), 'Induction of Decision Trees', *Machine Learning* **1**, pp.81—106.

Rada, R. (1985), Gradualness facilitates knowledge refinement, *in* 'IEEE Transactions on Pattern Analysis and Machine Intelligence', 7, 5, pp. 523--530.

Rainsford, D. & Mackaness, W. (2002), Template matching in support of generalization of rural buildings', in *Proc. Of Advances in Spatial Data Handling (SDH 2002)*, Ottawa, Canada, pp. 137--151.

Raju, K.; Duckstein, L. & Arondel, C. (2000), 'Multicriterion analysis for sustainable water resources planning: A case study in Spain', *Water Resources Management* **14(6)**, pp. 435--456.

Regnauld, N. (2002), 'Généralisation des bâtiments', Généralisation et représentation multiple, dir. Anne Ruas, Hermès Lavoisier, chap. 14, pp. 257--272.

Regnauld, N. (2001), Constraint based mechanism to achieve automatic generalisation using an agent modelling, *in* '9th GISRUK conference', University of Glamorgan, pp. 329--332.

Regnauld, N. (1998), 'Généralisation du bâti: structure spatiale de type graphe et représentation cartographique', Thèse de l'université de Provence-Aix-Marseille I. Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/REGNAULD/These_Regnauld_1998.pdf

Regnauld, N. & McMaster, RB. (2007), A Synoptic View of Generalisation Operators, *in*: WA Mackaness, A Ruas, and LT Sarjakowski (eds), Generalisation of Geographic Information: Cartographic Modelling and Applications. Oxford, UK: Elsevier, pp. 37--66.

Reichenbacher, T. (1995), Knowledge acquisition in map generalization using interactive systems and machine learning, *in* 'Proceeding of the 17th International Cartographic Conference', Barcelona, vol. 12, pp. 2221--2230.

Reynolds, S.I. (2000), Adaptive Resolution Model-Free Reinforcement Learning: Decision Boundary Partitioning, *in* 'Proceedings of the Seventeenth International Conference on Machine Learning', Morgan Kaufmann, pp. 783--790.

Ribeiro, M. & Wassermann, R. (2006), First steps towards revising ontologies, *in* 'Proceedings of WONRO', 11 pages.

Rieger, M. K. & Coulson, M. R. (1993), Consensus or confusion: cartographers' knowledge of generalisation, *Cartographica* **30**(23), pp. 69--80.

Roth-Berghofer, T. & Iglezakis, I. (2001), Six Steps in Case-Based Reasoning:Towards a Maintenance Methodology for Case-Based Reasoning Systems. *in* 'Proceedings of the 9th German Workshop on Case-Based Reasoning,', Baden-Baden, Germany, pp. 198--208.

Roy B. (1997), Un chaînon manquant en RO-AD : les conclusions robustes, Université Paris-Dauphine, Cahier du LAMSADE n° 144 ; version anglaise : A missing link in OR-DA: Robustness analysis, *Foundations of Computing and Decision Sciences* **23(3)**, pp.141--160.

Roy, B. (1991), 'The outranking approach and the foundations of ELECTRE methods', *Theory and Decision* **31**, pp. 141--160.

Roy, B. (1985), *Méthodologie multicritère d'aide à la décision*, Economica.

Roy, N. & McCallum, A. (2001), Toward optimal active learning through sampling estimation of error reduction, *in* 'Proceedings of the 18th International Conf. on Machine Learning', pp. 441--448.

Ruas, A. (2002), 'Les problématiques de l'automatisation de la généralisation', Généralisation et représentation multiple, dir. Anne Ruas, Hermès Lavoisier, chap. 4, pp. 75--90.

Ruas, A. (1999), 'Modèle de généralisation de données géographiques a base de contraintes et d'autonomie', Thèse de l'université de Marne la Vallée, laboratoire COGIT. Disponible sur le site web de l'IGN :
http://recherche.ign.fr/labos/cogit/pdf/THESES/RUAS/These_Ruas_1999.zip

Ruas, A. & Bianchin, A. (2002), 'Echelle et niveau de détail', Généralisation et représentation multiple, dir. Anne Ruas, Hermès Lavoisier, chap. 1, pp. 25--44.

Ruas, A.; Dyevre, A.; Duchêne, D. & Taillandier, P. (2006), Methods for improving and updating the knowledge of a generalization system, *in* 'Autocarto', Vancouver, Disponible sur le site web :
http://www.cartogis.org/publications/autocarto-2006/ruasdyevreduchenetaillandier.pdf/download

Ruas, A. & Duchêne, C. (2007), A prototype generalisation system based on the multi-agent system paradigm, *in* W.A. Mackaness; A. Ruas & L.T. Sarjakoski, ed.,'Generalisation of Geographic information: cartographic modelling and applications', Elsevier Ltd, pp. 269--284.

Ruas, A. & Holzapfel, F. (2003), Automatic characterisation of building alignments by means of expert knowledge, *in* '21st International Cartographic Conference', Durban, South Africa, pp. 1604--1615.

Ruas, A. & Plazanet, C. (1996), strategies for automated generalization, *in* '7th international symposium on spatial data handling', pp. 319--336.

Russel, S. & Norvig, P. (2006), *Intelligence artificielle*, Pearson Education.

Saux, E.; Thibaud, R.; Devogele, T.; Béra, R. & Guilbert, E. (2002), 'La généralisation des cartes marines', Généralisation et représentation multiple, dir. Anne Ruas, Hermès Lavoisier, pp. 303--317.

Schylberg, L. (1992), Cartographic amalgamation of area objects, *in* 'Proceeding of ISPRS'92', pp. 135 --138.

Sester, M. (2005), 'Optimization approaches for generalisation and data abstraction', *International Journal of Geographical Information Science* **19**(8--9), pp. 871--897.

Sester, M. (2000), 'Generalization based on least square adjustment', *International archives of photogrammetry and remote sensing* **23, part B4**, pp. 931--938.

Sester, M. (1998), Interpretation of Spatial Data Bases using Machine Learning Techniques, *in* 'Proceedings of the 8th International Symposium on Spatial Data Handling', Vancouver, Canada, pp. 88--97.

Shafer, G. (1976), *A mathematical theory of evidence*, Princeton University Press.

Shea, K. & McMaster, R. (1989), Cartographic generalization in a digital environment: When and how to generalize, *in* 'AutoCarto 9', pp. 56--67.

Shepherd, A. (1997), *Second-Order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-Layer Perceptrons*, Springer-Verlag London.

Shoham, Y. (1993), 'Agent-oriented programming', *Artificial Intelligence* **60**, p.51--92.

Shortliffe, E.H. (1976), *Computer-Based Medical Consultation:MYCIN*, American Elsevier.
Singh, A.; Nowak, R. & Ramanathan, P., (2006), Active learning for adaptive mobile sensing networks, *in* 'Proceedings of the fifth international conference on Information processing in sensor networks'.

Smets, P. (1990), 'Constructing the pignistic probability function in a context of uncertainty'. *Uncertainty in Artificial Intelligence* **5**, pp. 29--39.

Smets, P. & Kennes, R. (1994), 'The transferable belief model', *Artificial Intelligence* **66(2)**, pp. 191--234.

Smith, V. (2003), Generalization for Medium Scale Mapping: Results and Statistics from One Production Implementation, *in* '5th ICA Workshop on progress in automated map generalisation'.

Smith, R. & Davis, R. (1981), 'Frameworks for Cooperation in Distributed Problem solving', *IEEE Transactions on Systems, Man and Cybernetics* **11**, pp. 61--70.

Smith, R. G.; Winston, H. A.; Mitchell, T. M. & Buchanan, B. G. (1985), Representation and use of explicit justifications for knowledge base refinement, *in* 'Proceedings of the Ninth International Joint Conference on Artificial Intelligence', pp. 673--680.

Solnon, C. & S., F. (2004), 'A study of ACO capabilities for solving the maximum clique problem', *Journal of Heuristics* **12(3)**, pp. 155--180.

Sorg-Madsen, N.; Thomsen, C. & Pena, J. (2003), Unsupervised feature subset selection, *in* 'Proceedings of the Workshop on Probabilistic Graphical Models for Classification', pp. 71--82.

Spronk, J., (1981), *Interactive multiple goal programming*, The Hage: Martinus Nijhoff.

Steiniger, S.; Lange, T.; Burghardt, D. & Weibel, R. (2008), 'An approach for the classification of urban building structures based on discriminant analysis techniques', *Transactions in GIS* **12(1)**, pp. 31--59.

Stone, M. (1974), 'Cross-validatory choice and assesment of statistical predictions', *Journal of the Royal Statistical Society Series B* **36**, pp. 111--147.

Suganthan, P. N.; Hansen, N.; Liang, J. J.; Deb, K.; Chen, J.; Auger, A. & S., T. (2005), 'Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization', Technical report, Nanyang Technological University.

Sutton, R. (1988), 'Learning to predict by the method of temporal differences', *Machine Learning* **3**, pp. 9--44.

Taillard, E. (2005), 'Few guidelines for analyzing methods', MIC2005: The Sixth Metaheuristics International Conference, Vienna, Austria, Tutorial.

Touya, G. (2008), First Thoughts for the Orchestration of Generalisation Methods on Heterogeneous Landscapes, *in* 'workshop on generalisation and multiple representation', Montpellier, France, Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/montpellier2008/papers/01_Touya.pdf

University of Edinburgh, IGN (1999), 'Measures on meso level and organisation'(DC3), Technical report, AGENT consortium.

Valiant, L. (1984), 'A theory of the learnable', *Communications of the ACM* **27(11)**, pp. 1134--1142.

Vapnik, V. (1995), *The nature of statistical learning theory*, Springer-Verlag.

Vincke, P. (1989), *L'aide multicritère à la décision*, Éditions de l'Université de Bruxelles.

Ware, J. M. & Jones, C. B. (1998), 'Conflict reduction in map using iterative improvement', *GeoInformatica* **2**(4), pp. 383--407.

Ware, J. M.; Jones, C. B. & Thomas, N. (2003), 'Automated map generalization with multiple operators: a simulated annealing approach', *international journal of geographical information sciences* **17**(8), pp. 743--769.

Watkins, C. & Dayan, P. (1992), 'Q-learning', *Machine Learning* **8**, pp. 279--292.

Webb, G. (1993), DLGref2: Techniques for Inductive Rule Refinement, *in* 'Proceedings of the 1993 IJCAI Workshop W16: Machine Learning and Knowledge Acquisition', pp. 236--252.

Weibel, R. (1996), A typology of constraints to line simplification, *in* 'Proceeding of the 7th International Symposium on Spatial Data Handling', SDH'96, Delft, The Netherlands, 9A1--9A14,

Weibel, R. & Dutton, G. (1999), Generalising spatial data and dealing with multiple representations, *in* P.A. Longley; M.F. Goodchild; D.J. Maguire & D.W. Rhind, ed.,'Geographical Information Systems: Principles, Techniques, Application and Management', Longman, pp. 125--155.

Weibel, R.; Keller, S. & Reichenbacher, T. (1995), Overcoming the Knowledge Acquisition Bottleneck in Map Generalization: the Role of Interactive Systems and Computational Intelligence, *in* '2nd COSIT conference', Semmering, Austria, pp. 139--156.

Weiss, G. (1999), *Multiagent systems. A modern approach to distributed artificial intelligence*, The MIT Press.

Wilson, I. D.; Ware, J. M. & A., W. J. (2003), 'Reducing graphic conflict in scale reduced maps using a genetic algorithm', *fifth workshop on progress in automated map generalisation*, Paris, France. Disponible sur le site web de la commission en généralisation de l'ACI : http://aci.ign.fr/BDpubli/paris2003/papers/wilson_et_al_v0.pdf

Witten, I. & Frank, (2005), *Data Mining: Practical Machine Learning Tools and Techniques*, E.Kaufmann.

Wolpert, D. & Macready, W. (1995), 'No Free-Lunch Theorems for Search', Technical report, Santa Fee Institute.

Yan, H. & Li, Z. (2004), A Voronoi-based algorithm for point cluster generalization, *in* 'Proceedings of 11th International Conference on Geometry and Graphics'**,** GuangZhou, China, CD-ROM.

Yu, W. (1992), 'Aide multicritère à la décision dans le cadre de la problématique du tri : Concepts, méthodes et applications', Thèse de l'université Paris Dauphine.

Publications

Ruas, A.; Dyevre, A.; Duchêne, D. & Taillandier, P. (2006), Methods for improving and updating the knowledge of a generalization system, *in* 'Autocarto', Vancouver, Disponible sur le site web :
http://www.cartogis.org/publications/autocarto-2006/ruasdyevreduchenetaillandier.pdf/download

Steiniger, S. & Taillandier, P. (2007), Improving Map Generalisation of Buildings by Introduction of Urban Context Rules, *in* 'Geocomputation', Maynooth, Ireland.

Taillandier, P. (2008), Knowledge diagnosis in systems based on an informed tree search strategy: application to cartographic generalisation, *in* 'CSTST Student Workshop', Cergy-Pontoise (Paris), France, pp. 589--594.

Taillandier, P. (2007), Automatic Knowledge Revision of a Generalisation System, *in* 'workshop on generalisation and multiple representation'. Moscow, Russia, Disponible sur le site web de la commission en généralisation de l'ACI :
http://aci.ign.fr/BDpubli/moscow2007/Taillandier_ICAWorkshop.pdf

Taillandier, P. (2007), Acquisition automatique de connaissances de guidage d'un processus de généralisation de données géographiques, *in* '8$^{\text{ième}}$ rencontres des jeunes chercheurs en intelligence artificielle (RJCIA)', Grenoble, pp. 213--229.

Taillandier, P. (2007), Révision à base d'agents des connaissances de guidage d'un processus de généralisation de données géographiques, *in* 'Journées Francophones sur les systèmes Multi-Agents (JFSMA)', Carcassonne, pp. 243--252.

Taillandier, P. (2007), 'Révision automatique des connaissances d'un processus de généralisation de données géographiques', *Le monde des cartes, bulletin trimestriel du comité français de cartographie* **194**, pp.65--75.

Taillandier, P.; Duchêne, C. & Drogoul, A. (2008), Knowledge revision in systems based on an informed tree search strategy: application to cartographic generalisation, *in* 'International Conference on Soft Computing as Transdisciplinary Science and Technology'. Cergy-Pontoise (Paris), France, pp. 273--278.