Modélisation et gestion des contraintes pour un problème d'optimisation sur-contraint : Application à l'aide à la décision pour la gestion du risque de ruissellement

Wassim Jaziri

Institut National des Sciences Appliquées de Rouen

THÈSE DE DOCTORAT

Spécialité : Informatique

Modélisation et gestion des contraintes pour un problème d'optimisation sur-contraint : Application à l'aide à la décision pour la gestion du risque de ruissellement

Présentée par :

Wassim JAZIRI

Le 09 juillet 2004

Devant le jury composé de :

Michel	Mainguenaud	Directeur de Thèse	Professeur à l'INSA de Rouen, France
Thierry	Paquet	Encadrant	Professeur à l'Université de Rouen, France
Adel	Alimi	Co-encadrant	Professeur à l'Université de Sfax, Tunisie
Jean-Paul	Cheylan	Président	Directeur de recherche CNRS, CIRAD, Montpellier, France
Robert	Jeansoulin	Rapporteur	Chargé de Recherche CNRS, LSIS, Marseille, France
Joël	Colloc	Rapporteur	Directeur LIH, Le Havre, France
Daniel	Delahaye	Examinateur	Professeur à l'Université de Caen, France

Laboratoire Perception, Systèmes, Information

Résumé :

Les travaux présentés dans ce mémoire traitent du problème de la modélisation des contraintes et de l'interaction système-utilisateurs pour la résolution de problèmes complexes et sur-contraints. La satisfaction des contraintes des décideurs constitue un élément fondamental dans tout processus de décision, contraint la marge de manœuvre du système et la qualité de la réponse pouvant être apportée au problème étudié. Cependant, il semble que cet aspect soit peu défini, manque de modélisation et de fondements génériques de traitement et de gestion. L'intérêt reste souvent porté sur la démarche et les techniques de résolution plutôt que sur la modélisation du problème lui-même. Ce travail se focalise sur ce besoin de modélisation. Nous proposons une nouvelle démarche interactive de progression dans la résolution des problèmes d'optimisation sous contraintes, notamment en intégrant les décideurs dans la définition de leurs contraintes et dans la prise de décision finale afin de les aider à décider au lieu de décider à leur place. L'approche développée a été appliquée pour l'optimisation des assolements dans la prévention des risques de ruissellement touchant les territoires agricoles. Les simulations ont été effectuées sur le territoire du Pays de Caux au nord-ouest de la France.

Mots-clés : Modélisation des contraintes, optimisation, aide à la décision, risque de ruissellement.

Abstract:

Works presented in this thesis deal with the problem of constraints modelling and the system-users interaction to solve complex and over-constrained problems. The satisfaction of decision-maker's constraints constitutes a fundamental element in any decision process and constrains the margin of the system and the quality of solution that can be offered. However, it seems that this aspect received few propositions and lacks of modelling and generic treatment. The interest is often carried on the gait and the techniques of resolution rather than the modelling of the problem and it becomes necessary to pay more attention to fill this need of modelling. In this orientation, we propose a new interactive gait of progression in the resolution of the constraint satisfaction and optimisation problems, notably while integrating the decision-makers in the definition of their constraints and in making a final decision in order to help them to decide instead of deciding at their place. The developed approach was applied for the optimisation of crops assignment in order to minimize the streaming risks concerning agricultural territories. The results of simulation were obtained for the territory of Pays de Caux in the northwest of France.

Key words: Constraints modelling, optimisation, decision-aid, risk of streaming.

Remerciements

Ce travail de thèse a été réalisé dans le cadre d'une co-tutelle entre l'Institut National des Sciences Appliquées de Rouen (France) et l'Ecole Nationale d'Ingénieurs de Sfax (Tunisie) au sein du laboratoire Perception, Systèmes et Information (PSI) avec le soutien financier du Conseil Général de la Région de Haute Normandie (France).

Tout d'abord, je tiens à remercier vivement Michel Mainguenaud, mon directeur de thèse, Thierry Paquet mon encadrant et Adel Alimi mon co-encadrant pour leur aide et leur support inestimable durant les années de thèse et pour m'avoir appris tant de choses avec tant de pédagogie. Leur soutien, leur aide et leurs compétences ont permis que toute la période de la thèse se déroule dans un excellent climat espérant avoir encore longtemps droit à leurs conseils pragmatiques.

Ce travail n'aurait pu être réalisé sans les nombreux échanges qu'il m'a été donné d'avoir avec les membres du laboratoire IDEES-MTG (Rouen, France). Je voudrais témoigner ma reconnaissance tout particulièrement à David Gaillard, Patrice Langlois et Daniel Delahaye pour nos nombreuses discussions instructives et pour le vrai plaisir que j'ai trouvé en travaillant avec eux, en espérant que cette collaboration ne soit qu'un début.

Côté PSI, je remercie très sincèrement tous les membres de l'équipe avec lesquels j'ai partagé un bout de chemin au cours de ces années de thèse, avec une mention particulière pour Jean Pierre Pécuchet, Jaques Labiche, Yves Lecourtier, Edwige Pissaloux, Laurent Heutte, Abdel Ennaji, Marc Engel, Christèle Lecomte, Patrick Dumas, Ameur Bensefia, Hichem Sefion, Dominique Ménétrier et Laurence Savouray.

Je tiens également à adresser mes remerciements aux personnes qui me font l'honneur de participer à ce jury de thèse.

Enfin, je terminerai ces remerciements par mes proches à qui j'adresse toute ma gratitude pour leurs sacrifices et leurs innombrables encouragements.

Table des matières

Table des figures

INTRODUCTION GENERALE :

Positionnement Formel

***Collecte des informations** : Problèmes de plus en plus pluridisciplinaires et complexes*

Les problèmes auxquels sont confrontés les chercheurs sont de plus en plus pluridisciplinaires faisant intervenir des disciplines aussi complémentaires que concurrentes tels que l'informatique, la médecine, la géographie, l'agronomie etc. L'aspect pluridisciplinaire implique naturellement des vocabulaires différents d'expression et de communication qui sont souvent à l'origine des soucis d'incompréhension et de confusion entre les différents acteurs. Le bon déroulement du processus d'acquisition et d'échange d'informations entre les différents acteurs impose une cohérence des informations échangées et des termes employés pour l'expression de leurs idées et points de vues.
Par ailleurs, la propagation de l'information à travers des environnements plus ou moins hétérogènes implique souvent une perte de l'information due à la divergence sémantique entre le monde informel et formel ou ce que nous modélisons et ce que nous mettons en exécution. Cette hétérogénéité, due essentiellement à la multitude d'acteurs intervenants provenant de domaines différents et ayant des niveaux de perception souvent divergents, est de plus en plus source d'ambiguïté et de souci rendant ainsi délicate la tâche d'acquisition et de formalisation des objectifs et des contraintes de l'application.

A l'image de ces problèmes pluridisciplinaires touchant divers secteurs d'activités tels que la planification, le transport etc., l'aménagement du territoire n'échappe pas à la règle et constitue un milieu de grande diversité regroupant des acteurs de capacités cognitives et de vocabulaire d'expression différents. L'échange d'informations en présence du facteur humain, nécessite un vocabulaire commun de définition et de normalisation des termes employés afin de traduire fidèlement les besoins des décideurs et de préserver la cohérence et la complétude de l'information lors de son passage entre les différents acteurs intervenants dans le processus de prise de décision. L'élaboration d'un vocabulaire commun sert à cerner les divergences et à minimiser les pertes d'informations via un meilleur dialogue entre les différents acteurs. Ceci permet entre autres de mieux définir les contraintes et les objectifs de l'application et d'assurer ainsi le bon déroulement du processus de décision.

Compte tenu de ces exigences, le rôle de l'homme d'étude consiste à bien se positionner dans de telles conditions (environnement hétérogène et pluridisciplinaire, divergence entre le monde sémantique et numérique) et à préserver les différents aspects du problème pour une meilleure analyse et exploitation des informations manipulées.

***Analyse des informations** : Nécessité d'une modélisation générique des problèmes de décision*

La fiabilité de la modélisation du problème et l'efficacité du processus de décision dépendent largement du bon déroulement de la phase d'acquisition et de propagation de l'information afin d'assurer une traduction fidèle de tous les aspects de la problématique. La modélisation fidèle des différents aspects d'un problème de décision nécessite l'étude de la complexité des contraintes exigées et le développement du mécanisme adéquat pour assurer leur satisfaction tout en restant dans le cadre des objectifs visés par l'application. La détermination de la satisfiabilité du système : sous ou sur-contraint et de la marge de manœuvre de l'homme d'étude pour proposer des solutions satisfaisantes en un temps de calcul raisonnable, nécessite le recours à une analyse de la complexité du problème traité. Cette analyse intègre nécessairement l'évaluation de la complexité des contraintes prises en compte et de leur comportement au cours d'un processus de simulation afin de déterminer les traitements nécessaires pour leur satisfaction dans le cadre d'un processus d'optimisation sous contraintes.

Environnements de plus en plus évolutifs

Les processus de prise de décision concernant les environnements évolutifs et complexes sont souvent lents et tout au long du déroulement du processus, des événements extérieurs peuvent modifier les données du problème et influencer ainsi la prise de décision. Les connaissances sont donc susceptibles d'évoluer entre le moment de l'étude du problème et la mise en exécution des décisions prises. Cette évolution des connaissances peut causer l'irréalisme des solutions choisies par le processus de résolution et implique la nécessité de prévoir ces changements et de distinguer ainsi les contraintes évolutives de celles qui ne le sont pas. Les solutions proposées doivent être assez souples pour pouvoir s'adapter à d'éventuels changements des données du problème. Il convient également de déterminer les traitements nécessaires pour assurer la cohérence du système et le bon déroulement de la phase d'exploitation des informations manipulées.

***Exploitation des informations** : Besoin d'un outil informatique, besoin de simulation*

L'informatique est une discipline qui, grâce à l'évolution croissante de ses technologies, est devenue indispensable à de nombreuses activités de recherche dans des disciplines variées. Son intérêt ne cesse d'augmenter avec l'évolution que connaissent les supports et outils informatiques devenus mieux adaptés à satisfaire les contraintes de plus en plus fortes en temps et coût. Sa sollicitude a dépassé la simple informatisation de certains processus de gestion pour s'attaquer à des applications plus complexes relevant de plusieurs disciplines, telles que l'agronomie, la géographie, etc.

La conception d'outils informatiques pour l'analyse et la résolution des problèmes connaît un succès grandissant dans grand nombre de disciplines, notamment pour les problèmes interactifs ou la simulation des systèmes complexes. En effet, les systèmes réels sont de plus en plus complexes et difficiles à analyser et à résoudre. Les chercheurs font souvent recours à la technique de simulation qui s'impose comme un choix adéquat évitant le recours à des maquettes ou des mesures sur un système réel (coût élevé, difficultés d'instrumentation) ou encore l'utilisation des modèles analytiques (loin de la réalité et peu aptes à refléter toute la complexité d'un système). La simulation d'un système réel consiste à l'expérimenter sur un modèle en exécutant des scénarios possibles de son fonctionnement. Cette technique favorise principalement la compréhension et l'analyse des systèmes complexes mais n'assure pas à elle seule le processus de résolution du problème. La modélisation et le traitement des applications réelles nécessitent le recours à des formalismes spécifiques telles que l'optimisation, l'affectation sous contraintes ou l'aide à la décision. Ces disciplines sont plutôt complémentaires que concurrentes et les décideurs, suivant les objectifs qu'ils expriment et les contraintes qu'ils exigent, peuvent faire basculer le problème vers l'une ou l'autre de ces trois disciplines.

***Perspectives de notre travail** : Modélisation des contraintes*

La mise en place d'outils informatiques pour la simulation doit faire face à la complexité croissante des problèmes traités, due essentiellement à la multitude d'acteurs intervenants qui sont souvent à l'origine de contraintes diverses et nécessitant des traitements complexes. Les contraintes jouent un rôle fondamental en tant que clauses restrictives limitant la marge de

manœuvre du système et la qualité de la réponse pouvant être apportée à un problème donné. Cependant, cet aspect manque de modélisation et de fondements génériques de formulation et de traitement. L'intérêt reste souvent porté sur la démarche et techniques de résolution plutôt que sur la modélisation du problème lui-même et il s'avère important d'accorder plus d'attention à cette tâche de modélisation. La satisfaction des contraintes s'impose donc comme étant un élément fondamental méritant plus d'intérêt et une meilleure modélisation par l'homme d'étude afin de mieux définir et gérer la complexité des problèmes de décision.

C'est dans cette perspective, que se situe notre travail qui consiste à modéliser le système de contraintes afin d'apporter des éléments de réponse aux problèmes complexes ou sur-contraints et de concevoir une démarche de progression dans la résolution du problème, notamment en intégrant les décideurs dans un processus interactif et itératif de définition et de négociation des contraintes. Notre intérêt s'est porté essentiellement sur la proposition d'une approche générique assurant à la fois le processus d'acquisition, de formalisation, de négociation et de satisfaction des contraintes dans le cadre d'un processus d'aide à la décision.

Projet de coopération pluridisciplinaire

Le présent travail allie étroitement des chercheurs de disciplines variées : Informatique (PSI[1]), Agronomie (INRA- INAPG[2]) et Géographie (IDEES- MTG[3]) qui collaborent dans le cadre de deux projets PNSIG (Programme National sur les SIG) et GESSOL 2000 (GEStion du SOL) du ministère de l'agriculture. La mise en œuvre de cette collaboration s'articule autour de la gestion des problèmes de risques de ruissellement touchant les territoires agricoles.
L'équipe des géographes est mobilisée pour recenser, à partir d'enquêtes exhaustives auprès des agriculteurs exploitants, les connaissances spatiales et économiques du terrain et les stratégies décisionnelles qui y sont appliquées.

[1] FRE CNRS 2645 PSI : Perception, Systèmes, Information, Université et INSA de Rouen. Site Web : http://psiserver.insa-rouen.fr/psi//.
[2] UMR 1048 INRA-INAPG SADAPT : Institut National de Recherche Agronomique, Institut National Agronomique Paris-Grignon, Systèmes Agraires et Développement : Activités, Produits, Territoires, centre de recherche de Versailles. Site Web : http://www.versailles.inra.fr/.
[3] UPRESA CNRS 6063 IDEES-MTG : Identités et Différenciation des Espaces de l'Environnement et des Sociétés, Modélisation et Traitement Graphique en Géographie, département de Géographie, UFR Lettres et Sciences Humaines, Université de Rouen. Site Web : http://www.univ-rouen.fr/MTG/.

Ayant déjà une certaine expérience de travail sur les problèmes de ruissellement et d'érosion, la contribution des agronomes consiste à apporter une connaissance du métier et des pratiques afin de pouvoir identifier et caractériser l'ensemble des phénomènes physiques et agronomiques intervenants dans la propagation du risque.

La tâche du PSI se situe dans la contribution au développement de nouvelles méthodes pour aider les décideurs à mieux gérer leurs espaces afin de contrôler les problèmes collectifs tout en satisfaisant leurs besoins individuels.

Plan du mémoire

Ce mémoire comporte cinq chapitres :

Le premier chapitre est une introduction à l'environnement applicatif qui motive notre travail. Nous exposons la problématique de la gestion collective du risque de ruissellement dans les territoires agricoles gérés par des acteurs individuels.

Le deuxième chapitre est consacré à l'étude des disciplines d'optimisation, d'affectation sous contraintes et d'aide à la décision et fait un tour d'horizon des principales méthodes de résolution des problèmes s'inscrivant dans l'une ou l'autre de ces disciplines, à savoir les méthodes approchées, exactes et les techniques d'analyse multicritère.

Dans le troisième chapitre, nous développons un aspect statique lié à l'expression des besoins des décideurs et à la formulation et la modélisation de ces besoins sous forme d'objectifs et de contraintes. La modélisation des objectifs permet de définir le cadre mono ou multicritère du problème et de déduire ainsi les méthodes classiques ou multicritères appropriées pour la résolution. La modélisation des contraintes permet de définir différentes classifications visant divers niveaux d'expertises et permettant d'analyser la complexité des contraintes et leur comportement au cours d'un processus de simulation.

Le quatrième chapitre s'attache à un aspect dynamique comportant une phase de négociation des contraintes des décideurs et une phase de formulation et de suivi de l'évolution de la satisfaction des contraintes au cours d'un processus d'optimisation.

Le cinquième chapitre est dédié à une présentation du système de simulation développé et son expérimentation sur un cas réel d'optimisation du risque de ruissellement dans le Pays de Caux (Seine Maritime, France).

CHAPITRE 1 :

Problématique métier et environnement applicatif

Sommaire

Notre travail s'articule autour de la gestion des processus de risques naturels et de l'analyse de l'influence des pratiques agricoles sur le déclenchement et la propagation de ces processus dans les territoires agricoles.

Ce chapitre introductif contient un exposé du cadre applicatif de notre travail. Nous présentons tout d'abord la problématique métier qui motive notre étude et nous analysons le problème étudié à travers trois logiques mises en œuvre sur le terrain : la logique décisionnelle, hydrologique et comportementale. Nous distinguons ensuite trois types de connaissances observées dans le domaine métier, à savoir les connaissances spatiales, économiques et agronomiques. Nous présentons enfin un bref aperçu sur les applications pratiques de lutte contre les risques naturels avant de conclure sur le positionnement formel de notre travail.

I. Problématique métier

La lutte contre les catastrophes naturelles touchant les territoires agricoles est devenue une préoccupation importante dans le domaine de l'aménagement du territoire. Les risques liés à l'eau sont certainement les plus fréquents et les plus dévastateurs sur les plans humain et matériel et continuent à prendre des proportions considérables notamment dans les pays du bassin méditerranéen[4]. Ils se manifestent par des processus de ruissellement, d'inondations et d'érosion qui sont souvent à l'origine d'événements chroniques et parfois catastrophiques dont sont victimes les collectivités et les populations situées dans les zones sensibles des territoires touchés (figure 1). Les dégâts observés se traduisent par des inondations boueuses des habitats, le ravinement des parcelles agricoles, la pollution des captages d'alimentation en eau potable par les particules en suspension, la destruction des routes et des biens, la réduction de la superficie des sols agricoles, la désertification du milieu naturel, etc.

[4] Selon [BOU KHEIR 01], des études de l'organisation des Nations Unies pour l'alimentation et l'agriculture (FAO : http://www.fao.org) ont montré qu'en Tunisie, 45% de la superficie du pays est affectée par l'érosion hydrique, au Maroc 40%, en Turquie 50% [CELIK 96], en Grèce 35%, et en Algérie 45% [CHEBBANI 99].

Figure 1 : Illustration de la gravité des dégâts causés par les processus d'inondations touchant les collectivités, Source : Institut Français de l'Environnement (IFEN : http://www.ifen.fr).

A ce titre, des inondations se manifestent de plus en plus régulièrement dans les petits vallons secs (figure 2) du Pays de Caux [DELAHAYE 95], l'une des régions les plus touchées en Haute Normandie (figure 3), dans le nord-ouest de la France [GARY 93] et qui motive notre étude.

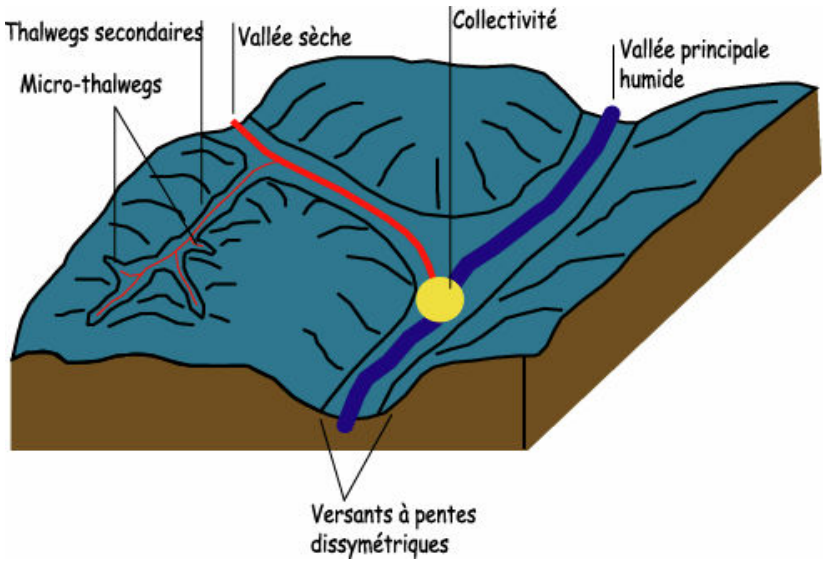


Figure 2 : Organisation générale du relief et des vallons secs dans le Pays de Caux.

1993-1996	Inondations	Mouvements de terrains (hors ceux consécutifs à la sécheresse)	Mouvements de terrain (consécutifs à la sécheresse)	Autres (avalanches, séismes...)	TOTAL Nombre	TOTAL Pour 100 communes de la région
Alsace	52	-	1	-	53	6
Aquitaine	325	14	141	-	480	21
Auvergne	86	1	6	-	93	7
Basse-Normandie	282	1	1	-	285	16
Bourgogne	105	2	4	-	111	5
Bretagne	290	2	-	-	292	23
Centre	86	10	107	-	203	11
Champagne-Ardenne	438	-	-	-	438	23
Corse	110	-	-	-	110	31
Franche-Comté	44	-	-	-	45	3
Haute-Normandie	315	23	1	-	339	24
Ile-de-France	171	5	90	-	266	21
Languedoc-Roussillon	389	1	10	-	400	26
Limousin	35	6	2	-	44	6
Lorraine	426	2	5	-	432	19
Midi-Pyrénées	268	20	209	1	498	17
Nord - Pas-de-Calais	511	18	40	5	574	37
Pays de la Loire	246	5	10	-	261	17
Picardie	451	1	2	-	454	20
Poitou-Charentes	391	2	43	-	436	30
Provence-Alpes-Côte d'Azur	266	48	21	4	339	35
Rhône-Alpes	522	40	6	22	590	21
France métropolitaine	**5 812**	**203**	**699**	**31**	**6 745**	**18**

Figure 3 : Nombre de communes ayant fait l'objet d'un arrêté de catastrophe naturelle entre 1993 et 1996. Source : Institut Français de l'Environnement (IFEN).

Nous présentons dans ce qui suit la description géophysique des phénomènes de ruissellement et d'érosion, le rôle des occupations de sol dans l'aggravation de ces processus et les solutions existantes pour faire face à ces risques.

I.1. Description géophysique des phénomènes de ruissellement et d'érosion

Le ruissellement est l'apparition d'écoulement de surface sur les parcelles du fait de la faible infiltration de l'eau dans le sol. Cette infiltration dépend du type de la culture pratiquée (culture d'hiver, de printemps) et de la nature du sol (sols limoneux,...) qui sont d'autant plus aggravants que les parcelles sont grandes. En effet, à partir d'un certain seuil de pluie, le sol n'absorbe plus l'eau qui commence à ruisseler selon la pente naturelle pour former le ruissellement en nappe. L'eau issue du ruissellement concentré est à l'origine des coulées boueuses affectant les zones urbanisées en fond de thalweg (figure 2) lorsque cet écoulement ne rencontre aucun obstacle susceptible de stopper sa progression ou de réduire son volume [LANGLOIS 02].

En fonction des conditions de pente et de sol, le ruissellement peut contribuer au décapage de la surface du sol formant ainsi l'érosion en nappe. Par concentration successive, l'érosion (pertes en terre) apparaît à partir du moment où la force de l'eau permet une dégradation significative de la surface des sols, formant ainsi des coulées boueuses destructrices [LECHEVALIER 91]. L'érosion hydrique correspond donc à la dissociation et au transport des particules sous l'action, en général combinée, de la pluie et du ruissellement. La quantité de sol exportée dépend des forces exercées sur le sol et de sa résistance face à l'érosion [AUZET 87]. Dans les régions de grandes cultures, comme le Pays de Caux par exemple, même une pluie de faible intensité[5] peut déclencher un ruissellement et une érosion importants.

I.2. Aggravation des phénomènes de risque

Dans la ligne de réflexion ayant conduit à notre démarche, nous prenons de plus en plus conscience du rôle joué par les occupations du sol dans la modélisation des risques. En effet, déclenchés par les pluies importantes, les risques deviennent plus graves en fonction de l'implantation humaine, de la morphologie du terrain et de l'activité agricole pratiquée. Les études menées dans ce cadre ont montré que l'évolution des pratiques agricoles joue un rôle déterminant dans le déclenchement et la propagation des processus de risque [BOIFFIN 88] [PAPY 91]. Les résultats des travaux menés dans [JAZIRI 02a][LANGLOIS 02][MARTIN 97][PAQUET 01] confirment cette pensée.

L'effet des pratiques culturales sur le ruissellement étant fonction de leur positionnement dans le temps (période et année) et dans l'espace (parcelle), une attention particulière doit être accordée à l'organisation des occupations du sol sur l'ensemble du territoire ainsi qu'aux successions culturales sur chaque parcelle [GAILLARD 02]. Une voie prometteuse pour la prévention des risques consiste donc à explorer les possibilités d'aménagements culturaux par un meilleur contrôle des décisions d'assolements sur les parcelles du territoire, dans le temps et dans l'espace.

I.3. Solutions classiques et leurs limites

Certaines stratégies d'actions ont été mises en place afin de mieux gérer les incidents provoqués par les excès du ruissellement de surface [DELAHAYE 92]. Les mesures prises

[5] Intensité parfois inférieure à 10 mm/h.

par les différents organismes concernés (exploitants, communes, instances nationales et régionales) visent à restreindre ou contrôler le ruissellement et l'érosion qui s'en suit afin de minimiser les dégâts. Actuellement, ces mesures restent limitées à des initiatives restreintes dans le temps et dans l'espace. L'aménagement des parties amont se limite trop souvent à des initiatives individuelles visant à réduire les nuisances dans le domaine d'intervention de l'acteur concerné et non sur l'ensemble de la collectivité. En aval des bassins, il existe des grands ouvrages d'écrêtage de crue qui sont coûteux en entretien et qui ne résolvent en rien les problèmes des communes situées en amont. Les bassins[6] de retenue d'eau installés en amont des zones les plus menacées et de part leur coût élevé d'installation et d'entretien (curage), ne sont efficaces que pour des événements d'une période de retour relativement brève.

Ce type d'aménagement développé pour remédier aux problèmes de risques naturels ne résout donc qu'en partie les problèmes d'inondations et présente des risques de sous-dimensionnement lors d'évènements exceptionnels [BERCHER 98]. Les solutions traditionnelles manquent d'efficacité du fait de leur nature correctrice (post-traitante) agissant après le déclenchement des catastrophes pour limiter leurs effets et traduisent le manque de collaboration des décideurs locaux envers les problèmes communs de risque.

Bien que certains programmes d'actions prônant des aménagements curatifs[7] et/ou préventifs soient engagés, notamment lors des périodes de crise, par les services et administrations compétents en la matière, les actions d'aménagements restent insuffisantes. Cette insuffisance est due à leur aspect embryonnaire et fragmentaire à cause de l'absence de continuité dans le raisonnement spatial et du manque de coordination et de concertation inter-acteurs et inter-services. Il s'avère ainsi nécessaire d'assurer une gestion plus efficace de l'ensemble des ressources communes (parcelles) afin de mieux gérer les processus de risque.

Cette volonté manifeste d'une gestion différente, préventive et globalisante à l'échelle de l'ensemble du territoire nécessite une analyse profonde du problème afin de déterminer l'origine de ces catastrophes et développer des solutions adéquates.

[6] Un bassin de retenue est un ouvrage de stockage destiné à retenir provisoirement l'eau de la pluie afin de réduire la quantité d'eau transportée sous l'effet de l'érosion hydrique.

[7] Ouvrages de retenue d'eau, recherche de nouveaux captages, etc.

II. Analyse du problème

Les enquêtes élaborées auprès des agriculteurs par les équipes des géographes de IDEES-MTG [DELAHAYE 99a][DELAHAYE 99b] et d'agronomes de l'INRA-INAPG [BLANCHARD 99][MARTIN 01] ont permis de dégager trois logiques : décisionnelle, hydrologique et comportementale mises en œuvre sur le terrain.

II.1. Logique décisionnelle

Elle concerne les facteurs qui interviennent et influencent la prise de décision sur les parcelles du territoire et les pratiques agricoles qui y sont appliquées. La logique décisionnelle permet de comprendre le comportement de chaque agriculteur et les facteurs limitant ses potentialités d'intervention sur ses parcelles.
En effet, la connaissance fine d'une exploitation agricole, et notamment ses marges de manœuvre en terme décisionnel, permet de proposer des actions adéquates conformément au diagnostic préalablement établit par les enquêtes menées.

II.1.1. Indépendance économique des exploitations

Les moyens d'actions sont entre les mains d'acteurs individuels, les exploitants, gérant chacun un espace fragmentaire formant l'exploitation. Une exploitation est une zone cultivée composée d'une ou plusieurs parcelles appartenant au même propriétaire (exploitant).
Chaque exploitant s'engage dans une stratégie agricole ayant comme objectif premier de mieux remplir ses besoins en production. Un objectif[8] de production s'exprime par la quantité à produire d'une culture dans une exploitation.

II.1.2. Pratiques agricoles appliquées

Les stratégies de production sont orientées en fonction de la situation du marché et de la rentabilité financière de chaque culture. De ce fait, les agriculteurs privilégient les cultures les plus rentables économiquement et ceci indépendamment de leur influence sur le risque.
En effet, comme nous allons le montrer dans ce qui suit, cette gestion individuelle et inefficace au niveau des unités décisionnelles (les exploitations) va moduler la gravité des processus de risque au niveau du bassin versant.

[8] Dans notre travail, nous considérons qu'un objectif de production peut aussi être exprimé par la surface à couvrir par une culture dans une exploitation.

II.2. Logique hydrologique

Elle concerne l'analyse des processus de risque et des connexions hydrauliques de voisinage entre les différentes zones du territoire. Ceci permet de comprendre l'activité de ruissellement et sa logique de propagation dans le territoire et de déterminer les zones les plus exposées au risque.

L'ensemble du territoire désigne une zone agronomique sensible constituée de plusieurs bassins versants contigus (figure 4). Un bassin versant est une surface hydrologiquement close, i.e., aucun écoulement n'y pénètre de l'extérieur et tous les excédents de précipitations s'évaporent ou s'écoulent par une seule section à l'exutoire [MUSY 03].

Les processus physiques de risque sont continus tout au long du bassin versant avec un transfert de risque entre les parcelles des différentes exploitations. A partir d'un certain seuil d'eau, l'ensemble du système se met à fonctionner avec un transfert complet de l'amont vers l'aval et le danger pour les collectivités situées à l'exutoire devient important car les surfaces d'alimentation sont parfois gigantesques [LANGLOIS 02]. Dans certains cas, à cause de la combinaison originale de la morphologie et de l'occupation du sol, les dégâts constatés dans les zones inondées sont dus à la quantité d'eau provenue de surfaces très éloignées des zones en question.

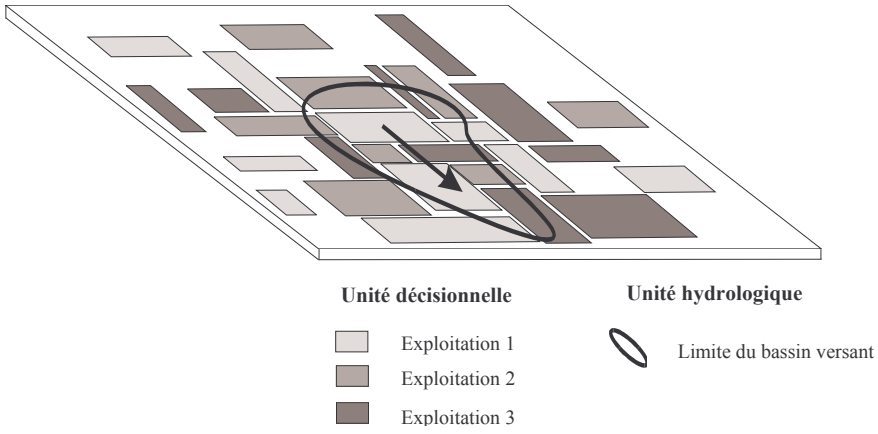


Figure 4 : Dissociation spatiale des unités décisionnelles (exploitations) et des unités de fonctionnement hydrologique (bassins versants).

Outre ces facteurs, les processus de péri-urbanisation/urbanisation contribuent de leur côté à l'aggravation du problème et la densification des infrastructures de transport accentue le problème en connectant des surfaces ruisselantes entre elles.

Malgré cette forte dépendance des exploitations vis-à-vis de la propagation du risque, la mobilité et l'élan de solidarité entre les exploitants restent très limités et leur attitude ne va pas dans le sens d'une démarche collective de lutte contre les problèmes communs. Ce dernier point sera détaillé dans la partie qui suit.

II.3. Logique comportementale et sociale

Cette logique permet de comprendre le comportement des agriculteurs et les relations humaines qu'ils peuvent entretenir entre eux et avec leur environnement (instances, élus locaux).

Malgré leur conscience des problèmes du ruissellement érosif, les exploitants cherchent avant tout à réduire les nuisances sur leurs propres parcelles et non sur l'ensemble de la collectivité. La mobilité et l'élan de solidarité entre eux sont très limités et leur inscription dans une démarche collective n'est pas évidente ce qui explique la difficulté qu'ils ont pour concevoir collectivement des solutions aux problèmes érosifs [GAILLARD 02].

Cette absence de solidarité s'explique d'une part par la mésentente entre certains exploitants (conflits d'intérêt divers) ce qui pousse ces derniers à émettre plusieurs réserves dans la coopération avec certains de leurs voisins. D'autre part, les entretiens réalisés par l'équipe MTG auprès des agriculteurs dans le bassin versant du Pays de Caux, ont permis de constater qu'une bonne partie des exploitants (environ 40% des agriculteurs enquêtés), n'avaient pas systématiquement conscience de la contribution de leurs pratiques sur leurs propres parcelles au fonctionnement global du processus de risque dans l'ensemble du bassin versant et aux dégâts subis par les parcelles des exploitations voisines.

Dans un tel contexte et afin d'aider le décideur dans sa tâche, il convient de résoudre le paradoxe risque collectif/objectifs individuels en tenant compte des moyens existants, de la réalité du terrain et de la divergence des comportements et intérêts individuels des exploitants. Notre tâche consiste à fournir au décideur des informations pertinentes et appropriées à ses problèmes, de nature à lui permettre de prendre des décisions éclairées pour mieux gérer l'espace de son travail au profil d'une gestion consciente et collective du risque de ruissellement.

Compte tenu de ces exigences et de la convergence entre la réalité décisionnelle, hydrologique et comportementale ainsi que la multitude de facteurs à prendre en compte, nous nous sommes orientés vers l'analyse des connaissances métier afin de mieux comprendre la problématique étudiée.

III. Connaissances métier

Dans l'analyse et l'évaluation des problèmes sur les risques de ruissellement, il est indispensable de tenir compte de façon simultanée, de facteurs divers et convergents à l'image des relations humaines, les activités pratiquées, les infrastructures et équipements utilisés, les lois et règlements, etc. De ce fait, l'apport des connaissances métier et des pratiques agricoles est crucial pour le processus d'aide à la décision relatif à la gestion du territoire. Nous présentons dans cette partie l'acquisition des connaissances spatiales, économiques et agronomiques par les experts métier ainsi que les contraintes du système et une synthèse sur la problématique.

III.1. Acquisition des connaissances

A l'issue des discussions élaborées avec les deux équipes de IDEES-MTG et de l'INRA-INAPG et de nombreuses visites sur le terrain d'étude, nous avons défini trois types de connaissances : spatiales, économiques et agronomiques.
Ayant déjà une certaine expérience de travail sur les problèmes de ruissellement et d'érosion, les équipes de géographes et d'agronomes sont mobilisées pour recenser, à partir d'enquêtes exhaustives auprès des agriculteurs exploitants, les connaissances du terrain et les pratiques qui y sont appliquées. Leur travail suit trois axes principaux : 1- un axe spatial concerne l'extraction et l'analyse des informations spatiales et géographiques des territoires agricoles ; 2- un axe économique relatif à l'analyse des situations financières des exploitants et à leurs stratégies de production ; 3- un axe agronomique lié à l'extraction des connaissances correspondant à la qualité du sol, aux règles d'affectation des cultures ainsi que les caractéristiques des cultures et leurs conduites dans le temps et dans l'espace.
Il faut noter que certaines connaissances métier sont susceptibles d'évoluer au cours du temps suite à des facteurs externes (climat, primes) et internes (dégradation du sol, stratégie de production adoptée).

III.1.1. Les connaissances spatiales

Ces connaissances aident à renseigner sur la répartition et l'organisation spatiale des parcelles entre les différentes exploitations, leur disposition et leurs relations de voisinage dans le bassin versant ainsi que la présence d'éléments paysagers notoires (haie, mares, talus, fossés). Elles sont extraites par les géographes (laboratoire IDEES-MTG) et via les Systèmes

d'Informations Géographiques [BURROUGH 86]. Un Système d'Information Géographique (SIG) est un système informatique permettant de manipuler et de gérer des informations à référence spatiale ou géographique contribuant notamment à la gestion de l'espace [DENEGRE 96]. Les fonctionnalités[9] d'un SIG permettent d'afficher et de localiser les parcelles, d'extraire les caractéristiques morphologiques du bassin versant, etc.

Chaque parcelle est identifiée par un ensemble de paramètres la caractérisant, à savoir, sa position dans le bassin, sa taille, sa pente, son exploitation et la surface en amont. Ce dernier paramètre reflète la surface des parcelles qui, par propagation de ruissellement, se déversent sur la parcelle considérée.
Les parcelles de la même exploitation peuvent être réparties suivant une organisation en bloc ou disséminées dans l'espace. Ceci détermine dès lors des possibilités d'intervention tout à fait différentes. Des propositions de découpages de certaines parcelles pourront aussi être envisagées suivant leur taille. Il faut naturellement vérifier une surface minimale de 3 hectares pour chaque sous-parcelle obtenue. Une dimension spatiale est également prise en compte suivant le type, avec ou sans élevage, d'exploitation agricole. Si l'activité d'élevage est présente dans l'exploitation et si la remise en herbe est proposée, la parcelle concernée ne doit pas se situer trop loin du corps de ferme qui constitue le point de référence [GAILLARD 02].

III.1.2. Les connaissances économiques

Elles concernent des connaissances relatives à la situation financière des exploitants et l'apport économique des différentes cultures. Chaque agriculteur oriente sa stratégie de production en fonction de la situation du marché et de la rentabilité financière de chaque culture pour satisfaire ses objectifs de production. De ce fait, les exploitants privilégient les cultures les plus rentables, en dépit d'autres cultures moins rentables ou pouvant être compensées par l'herbe ou par l'achat d'aliments extérieurs.

Les cultures se distinguent suivant leur rendement économique direct ou indirect. En effet, contrairement à certaines cultures, comme le blé, qui génèrent un rendement qui a directement une signification marchande (sont vendues directement), certaines cultures, comme le maïs, ont un effet indirect sur l'équilibre économique de l'exploitation du fait de leur participation au système d'élevage en assurant la nourriture des animaux.

[9] Différents outils de développement sont proposés, entre autres : ArcInfo, ArcView, ArcPAD, MapObjects, etc.

Un autre impératif, lié au précédent, concerne les systèmes de production qui ont tendance à se spécialiser ce qui implique un éventail des assolements pratiqués de plus en plus restreint. Certains agriculteurs se fixent une stratégie de production (un cycle d'assolement) à moyen ou à long terme et n'acceptent aucune modification de ses paramètres, notamment quand cette modification pourra représenter un surcoût dans leur schéma financier. A cet effet, certaines primes sont souvent accordées aux agriculteurs dans le cadre d'actions locales et régionales d'aménagement (bandes enherbées, inter-cultures, jachère faune sauvage, etc.) afin de les inciter à modifier leurs pratiques agricoles et à respecter les orientations stratégiques européennes.

III.1.3. Les connaissances agronomiques

Elles sont définies par les experts agronomes et concernent les caractéristiques des cultures et leurs logiques de succession, le comportement des sols et le calendrier des pratiques à entreprendre pour chaque période de l'année.

Chaque type de culture possède des exigences vis à vis de la qualité du sol, de la taille de la parcelle et du type de pente nécessaires à son implantation. Généralement, les cultures, à l'exception de l'herbe et du bois, doivent être localisées sur des parcelles de pente inférieure à 10 % puisque les valeurs de pente fortes interdisent l'utilisation des machines agricoles nécessaires pour cultiver ces productions.

L'exigence de taille apparaît en raison de l'intervention d'outils et machines de taille et encombrement importants. En effet, certaines cultures exigent des possibilités de manœuvre au bord des parcelles et l'intervention sur de grandes parcelles répond à cet impératif. Dans le même contexte, l'intensification des modes de production et notamment des outils et machines utilisées, nécessite de grandes surfaces pour manœuvrer. Le morcellement parcellaire multiplie les interventions sur des pièces éloignées et réparties sur le territoire de l'exploitation et ne permet pas de répondre aux objectifs de production et de productivité imposés par le système de production.

En ce qui concerne la qualité du sol, certaines cultures à l'image du lin, de la betterave et de la pomme de terre doivent bénéficier de conditions satisfaisantes pour pouvoir pousser d'autant plus que ces assolements engagent des coûts de productions assez élevés (machines, traitement).

A chaque parcelle, correspond un cycle d'assolement qui permet de respecter le rythme naturel des cultures et de préserver la qualité du sol pour les parcelles. Un cycle d'assolement

s'exprime en précédent / suivant[10] et dépend de la fréquence de retour des différents types de culture et des possibilités de succession entre elles[11]. La fréquence de retour des cultures détermine la période minimale qui doit séparer deux présences successives d'une culture sur la même parcelle. Cette interdiction de retours rapprochés sur la même parcelle s'explique par la nécessité de régénérer le sol avant une réimplantation.

Notre tâche consiste à modéliser ces connaissances spatiales, agraires et économiques relevées sur le bassin versant afin de proposer des solutions d'aménagement qui tiennent compte de la réalité hydrologique du terrain, de la logique économique de l'exploitation et des contraintes du système.

III.2. Contraintes du système et synthèse de la problématique

Le lien étroit entre les pratiques agricoles et la continuité du risque dans les bassins versants inondés est souvent mal analysé et justifie cette réflexion pour développer des outils d'analyse et de gestion des processus de risque. Ainsi, lorsque la décision individuelle de l'agriculteur sur ses parcelles peut avoir des conséquences sur les parcelles de ses voisins dans le bassin versant (transfert de risque), ceci impose une concertation entre agriculteurs et une coordination de leurs pratiques agricoles en faveur d'un comportement plus conscient envers les risques collectifs.

III.2.1. Contraintes du système

Les problèmes de gestion du risque de ruissellement intègrent généralement plusieurs acteurs qui interagissent et expriment des besoins et des exigences différentes. Il ressort de l'analyse des différents types de connaissances ci-dessus évoqués que lors de chaque décision prise ou de chaque intervention sur les parcelles du territoire, un ensemble de contraintes doit être pris en compte à plusieurs niveaux de l'espace :

- *Au niveau local de chaque parcelle* lors de l'affectation d'une culture à une parcelle. Les occupations du sol doivent respecter les contraintes agronomiques relatives aux valeurs de taille, de pente et de type de sol autorisées à leurs implantations.

[10] Précédent étant la culture de la période passée et suivant la culture de la période à suivre (année ou saison).
[11] Certaines cultures ne peuvent pas se succéder dans les rotations d'assolement.

- *Au niveau décisionnel de chaque exploitation* où un objectif de production doit être atteint pour chaque type de culture.

- *Au niveau global du bassin versant* où la valeur du risque de ruissellement mesurée à l'exutoire du bassin versant doit être minimale. L'exutoire d'un bassin désigne le point le plus en aval qui reçoit tout le volume d'eau venant des différentes zones du bassin.

III.2.2. Synthèse

Pour apporter des solutions efficaces de lutte contre les problèmes de risque, il s'avère judicieux de reconsidérer les règles d'assolement à l'échelon de l'exploitation individuelle au profit d'une gestion globale du ruissellement érosif à l'échelle du bassin versant. La parcelle doit être considérée comme appartenant à un bassin versant (unité hydraulique) et replacée dans le cadre de l'exploitation agricole qui représente son unité décisionnelle afin de mieux gérer les pratiques qui y sont appliquées et de contrôler leurs conséquences sur le risque pour l'ensemble de la collectivité.

Il est toutefois impératif d'intégrer les décideurs dans le processus de prise de décision et de les aider à exprimer eux-mêmes leurs besoins et contribuer à la résolution de leurs problèmes collectifs. Dans ce sens, déçus de n'avoir jamais été contactés par des spécialistes en matière de lutte contre le ruissellement, les agriculteurs ont accueilli notre démarche de manière positive et souhaitent avant tout ne pas être exclus de la concertation et des éventuelles prises de décision et avoir ainsi la parole pour exprimer leurs points de vues. Ils souhaitent que nous les aidions à décider plutôt que de décider à leur place et expriment leur disposition à modifier leurs pratiques culturales mais sous certaines conditions :

- Ne pas trop modifier leurs pratiques : beaucoup d'agriculteurs se sont engagés dans une stratégie agricole et ne veulent pas en modifier les paramètres.

- Les nouvelles pratiques ne doivent surtout pas représenter un surcoût dans leur schéma financier.

En effet, l'aménagement du territoire agricole étant en grande partie à l'initiative des exploitants, les surfaces cultivées étant en effet considérables, une voie prometteuse pour la prévention des risques consiste à simuler le comportement des exploitants dans l'exploration des possibilités d'aménagements « doux » favorisant la diminution du risque collectif de ruissellement au niveau du bassin versant.

Une large place doit être accordée à la concertation et à la recherche de solutions communes en tenant compte des interactions spatiales "multi-echelles" entre les différents compartiments de l'espace, à savoir le bassin versant, l'exploitation et la parcelle (figure 5).

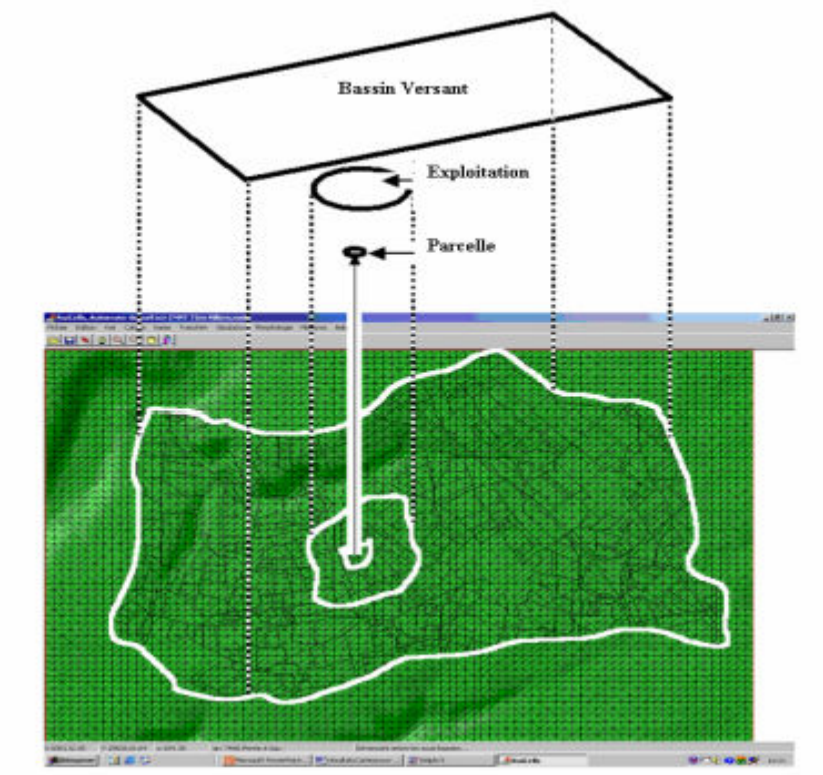


Figure 5 : Les différentes composantes de l'espace : parcelle, exploitation et bassin versant.

Nous espérons ainsi offrir aux agriculteurs et aux acteurs locaux la possibilité de maîtriser collectivement le ruissellement érosif par des modifications coordonnées des systèmes de cultures sans trop dégrader les satisfactions individuelles de production. Une attention toute particulière sera apportée aux techniques culturales afin d'identifier, par simulation et analyse de configurations, des solutions optimales de placement des cultures dans le temps et dans l'espace.

IV. Applications pratiques

Différentes réflexions ont été engagées pour comprendre, analyser et proposer des solutions de lutte contre les risques naturels. A ce sujet, Martin et al. ont étudié dans [MARTIN 97][MARTIN 98][MARTIN 99] les solutions de lutte contre le ruissellement dans un bassin versant notamment en proposant des interventions au niveau de la parcelle

(fragmentation, inter-culture) et de l'exploitation (l'organisation du travail et de la main d'œuvre).
Une autre étude menée par Le Ber et al. [LEBER 98a][LEBER 98b] s'est inspirée du modèle d'éco-résolution [BURA 91] pour résoudre un problème d'organisation des cultures dans un territoire agricole de façon à atteindre pour chacune, un volume de production recherché sans toutefois chercher à optimiser un objectif global de risque. Différentes approches à base de système multi-agents, de système expert et de recuit simulé ont été testées sur les terrains de Lignéville (228 parcelles) et Valleroy (210 parcelles) situés sur le plateau de Vittel (Région Lorraine, dans l'est de la France).

D'autres démarches ont été proposées, entre autres par Barthès [BARTHES 98], Benoît [BENOIT 98], Bousquet [BOUSQUET 01], Boli [BOLI 98], Celik [CELIK 96], Chebbani [CHEBBANI 99], Chen [CHEN 99], Delahaye [DELAHAYE 99c], Gallien [GALLIEN 95], Jetten [JETTEN 96], Langlois [LANGLOIS 02], Le Ber [LEBER 98c], Le Bissonnais [LE BISSONNAIS 96][LE BISSONNAIS 97], Legeard [LEGEARD 00], Martin [MARTIN 01], Roose [ROOSE 92], Tim [TIM 94].

Cependant, il semble que toutes ces démarches ne traitent qu'un nombre restreint de contraintes sans complexité majeure. En plus, bien que l'aménagement du territoire agricole soit en grande partie à l'initiative des exploitants, ces derniers sont rarement intégrés dans le processus de décision de ces travaux. L'homme d'étude se positionne souvent à la place des décideurs et joue donc le rôle de tous les acteurs en simulant leurs choix et comportements sans les intégrer effectivement dans le processus de résolution.
Ces approches ne comportent pas d'analyse conceptuelle globale et se limitent à l'analyse de l'aspect métier. Les analyses restent souvent limitées aux cas étudiés et ne tentent pas d'appréhender tous les aspects du problème afin de proposer un cadre générique vis à vis de la problématique traitée. Le vocabulaire utilisé dans ces démarches reste restreint au domaine de compétence de l'homme d'étude représentant souvent un agronome ou un géographe. Ces approches ressemblent ainsi plus à la simulation du raisonnement d'un expert qu'à un processus de décision basé sur la concertation entre les différents auteurs.

Bien qu'elles traitent la même catégorie de problèmes applicatifs, ces applications ne répondent pas à nos attentes et ne s'inscrivent pas dans la voie que nous voulons suivre, à savoir la conception d'un outil formel et générique d'optimisation, de satisfaction de contraintes et d'aide à la décision. Nous pensons ainsi que le caractère pluridisciplinaire de la

problématique est désormais admis. Il est en effet essentiel de sortir du simple cadre de l'ingénierie hydraulique ou agronomique classique, et d'aborder le problème sous un angle plus générique en accordant une large place à la modélisation d'outils informatiques. De tels outils visent à assurer une concertation entre les différents acteurs et une modélisation générique et fondée des contraintes du système afin de permettre la gestion de leur complexité conceptuelle et de simulation.

V. Conclusion

La gestion des risques liés à l'eau dans les terrains agricoles est une tâche complexe, en raison des multiples interactions qui existent entre les composantes du système hydrologique et économique d'une part, ainsi que les nombreuses catégories d'intervenants impliqués (agriculteurs, décideurs, instances européennes, agronomes, etc.) et les relations humaines qu'ils peuvent entretenir d'autre part.
Cette complexité, ainsi que la prise en compte d'un ensemble de contraintes plus ou moins complexes, font que les différentes activités touchant de près ou de loin à la gestion des problèmes communs entre des acteurs de profils et d'intérêts divergents doivent être abordées de manière plus efficace. Ces activités ne doivent plus être traitées de manière restreinte ou au cas par cas, mais selon une approche conceptuelle globale basée sur des fondements génériques modélisant fidèlement les différents aspects pertinents à la problématique.

Les travaux que nous présentons se réfèrent à une logique de prévention des risques et d'intégration des agriculteurs dans la prise de décision par la recherche de modalités d'intervention agissant avant le déclenchement des catastrophes et rendant compatibles la maîtrise du ruissellement et la satisfaction des besoins des agriculteurs. Ces besoins peuvent être exprimés suivant trois directions :

- Un besoin d'optimisation relatif à la minimisation du risque de ruissellement en adoptant une stratégie de gestion globale du bassin versant. Cette stratégie est basée sur un mécanisme permettant de montrer aux exploitants l'impact éventuel de leurs choix culturaux sur les parcelles voisines et sur l'ensemble du bassin versant.

- Un besoin de satisfaction des contraintes qui consiste à satisfaire les exigences des exploitants afin de garantir l'acceptation des solutions proposées. Il est à noter que les décideurs imposent souvent trop de contraintes plus ou moins complexes ce qui limite la marge de manœuvre du système et complique la tâche de l'homme d'étude. Nous envisageons

ainsi de procéder à une négociation entre le système et les décideurs pour relaxer certaines contraintes afin de rendre le système satisfiable (de complexité traitable).

- Un besoin d'aide à la décision relatif à l'intégration des décideurs dans le processus de prise de décision pour favoriser leur solidarité dans la lutte contre les risques collectifs.

A partir de là, il s'avère évident que le cadre de notre travail est un cadre mixte d'optimisation et de satisfaction de contraintes à l'intérieur d'un cadre plus large d'aide à la décision. Il nous semble en effet que dans un contexte de décision face au risque, nous pouvons tirer profit des trois disciplines afin de modéliser et de négocier les contraintes dans une approche multidisciplinaire d'optimisation du risque et d'intégration des décideurs dans la prise de décision.

Le chapitre suivant propose une étude formelle de l'état de l'art autour de ces différentes disciplines en présentant un tour d'horizon des principales méthodes et techniques de résolution utilisées.

CHAPITRE 2 :

Etat de l'art formel autour des cadres et techniques de résolution des problèmes

Sommaire

Comme nous l'avons vu dans le chapitre précédent, la gestion des risques liés à l'eau dans les territoires agricoles est une tâche complexe en raison des nombreuses catégories d'acteurs intervenants et de la convergence de leurs perceptions et points de vues. Ces acteurs, issus de domaines variés et ayant des vocabulaires d'expression différents, doivent souvent collaborer pour apporter des solutions efficaces aux problèmes collectifs de risque. Le problème doit être abordé suivant une démarche générique qui prend en compte simultanément l'objectif collectif de risque et les contraintes individuelles exigées par chaque agriculteur.

Trois cadres formels de traitement et de résolution de problèmes faisant intervenir des contraintes peuvent être distingués, à savoir l'optimisation, l'affectation sous contraintes et l'aide à la décision. Les décideurs, suivant les objectifs qu'ils expriment et les contraintes qu'ils exigent, peuvent orienter le problème vers l'une ou l'autre de ces disciplines.
De nombreuses méthodes de résolution ont été développées dans le cadre de ces trois disciplines dont les frontières restent jusqu'à lors arbitraires et mal définies [COURBON 93]. Nous distinguons notamment les méthodes approchées qui représentent les techniques classiques d'optimisation, les méthodes exactes et les approches multicritères. Bien que certaines techniques soient plus adaptées à l'un ou l'autre de ces domaines de recherche, les différentes méthodes de résolution peuvent être utilisées conjointement pour résoudre les problèmes s'inscrivant dans le cadre de ces disciplines d'autant plus qu'elles s'attaquent aux mêmes catégories de problèmes (figure 6).

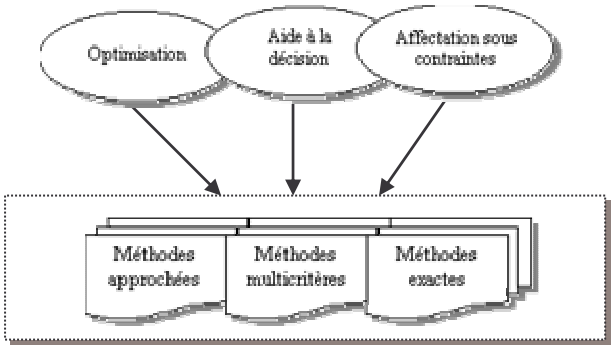


Figure 6 : L'utilisation des techniques de résolution des problèmes dans le cadre des différentes disciplines de traitement des contraintes.

La première partie de ce chapitre sera consacrée à l'étude des cadres formels d'optimisation, d'affectation sous contraintes et d'aide à la décision sans spécifier les techniques de résolution employées. Ensuite, nous faisons un tour d'horizon, non exhaustif, des principales méthodes de résolution des problèmes utilisées dans le cadre de ces différentes

disciplines, à savoir les méthodes approchées, exactes et les techniques d'analyse multicritère en nous intéressant plus particulièrement au fonctionnement des méthodes les plus adaptées à la résolution de ce travail. Ce chapitre se termine avec une synthèse sur la complémentarité des trois disciplines et sur l'apport de ce travail dans la formalisation et la modélisation des contraintes.

I. Cadres formels de résolution des problèmes

Les différents travaux traitant des applications similaires, notamment dans la planification [DUBOIS 95][EASTMAN 72][FOX 87][MINTON 92], la gestion des ressources humaines et naturelles [EL MAGNOUNI 93][GERSHON 84][MARTEL 93][ROY 92] [VIJAYALAKSHMI 87], le secteur médical [DEGOULET 91][MILLER 86][SEFION 02], le transport [BARNIER 02][GRANGER 02][LIVADAS 00][PRANDINI 99] et l'aménagement du territoire [JANSSEN 90][MAYSTRE 94][PEREIRA 93], ont été menés dans les domaines de l'optimisation, de l'affectation sous contraintes et de l'aide à la décision. Comme nous allons le montrer dans ce qui suit, ces disciplines sont plutôt complémentaires que concurrentes et s'intéressent aux mêmes domaines d'application.

I.1. L'optimisation

L'optimisation consiste à trouver une bonne solution répondant à des critères d'évaluations et satisfaisant un ensemble restreint de contraintes, en un temps de calcul raisonnable. Elle fait intervenir une fonction d'optimisation représentée souvent sous une forme mathématique censée traduire les objectifs du problème. L'expression des objectifs sous forme d'une fonction de coût ou de profit est une tâche délicate à cause de la divergence[12] entre la réalité et la simulation. Dans la pratique, les problèmes d'optimisation peuvent atteindre rapidement une grande complexité et nécessiter des temps de calcul considérables à cause de l'explosion combinatoire du nombre de solutions potentielles. Les méthodes approchées constituent une alternative très intéressante pour traiter ce genre de problèmes où la satisfaction des contraintes n'est pas primordiale.

[12] Les problèmes concrets sont en général beaucoup plus complexes que les cas théoriques.

Nous définissons tout d'abord l'optimisation combinatoire et multicritère. Nous présentons ensuite les conditions d'optimalité caractérisant l'existence d'un optimum avant de synthétiser le cadre de l'optimisation et ses limites.

I.1.1. L'optimisation combinatoire

[HAO 99] définit un problème d'optimisation combinatoire comme étant un ensemble d'instances (ensemble de valeurs pour les paramètres du problème). S étant l'ensemble des solutions du problème, une instance I d'un problème d'optimisation est un couple (X, f) où $X \subseteq S$ est un ensemble fini de solutions admissibles, et f une fonction objectif à optimiser $f : X \rightarrow R$ qui assigne à chaque solution $s \in X$ une valeur f(s). Résoudre un problème d'optimisation combinatoire (plus précisément une telle instance du problème) consiste à trouver une solution $s^* \in X$ optimisant la valeur de la fonction f, i.e., trouver $s^* \in X$ tel que f(s*) est meilleure que f(s) pour tout élément $s \in X$. Une telle solution s* s'appelle une solution optimale ou un optimum global.

La plupart des problèmes d'optimisation combinatoire appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données [GAREY 79]. Un problème est dit NP-difficile si aucun algorithme polynomial n'est connu pour le résoudre. Il est dit polynomial s'il existe un algorithme permettant de trouver une solution optimale pour toutes ses instances en un temps polynomial par rapport à la taille de l'instance. Pour la majorité des problèmes d'optimisation combinatoire, aucun algorithme polynomial n'est connu actuellement et leur difficulté intrinsèque est caractérisée par la théorie de la NP-complétude [GAREY 79].
La fonction d'optimisation comporte couramment un seul critère capable de déterminer l'optimum. Dans le cas contraire, l'objectif consiste à trouver un bon compromis en présence de critères multiples et le problème se ramène plutôt à un cas d'optimisation multicritère [OTHMANI 98].

I.1.2. L'optimisation multicritère

L'optimisation multicritère [SOLAND 79][STEUER 86] représente une extension à la notion d'optimisation classique dont le domaine se limite aux problèmes cherchant à améliorer un critère unique. Le choix de la meilleure solution se base sur l'évaluation parallèle de

plusieurs critères d'optimalité[13]. D'une façon générale, le traitement de l'aspect multicritère peut suivre trois orientations :

- Prendre en compte un seul critère (le plus important) et transformer les autres critères sous forme de contraintes et /ou d'heuristiques.

- Rassembler les différents critères dans la même fonction F ce qui revient à ramener les critères à un seul par sommation ou moyenne pondérée. Le choix des poids [VANSNICK 86] traduit l'importance de chaque critère.

- Prendre en compte les différents critères séparément et tenter de trouver un compromis vérifiant leur optimisation.

Cependant, dans quels cas peut-on vraiment parler d'optimisation et orienter ainsi la résolution vers la recherche de l'optimum?

I.1.3. Les conditions d'optimisation

Le cadre d'optimisation se révèle parfois inefficace face à des problèmes où aucun optimum ne peut être dégagé. En effet, l'existence de cet optimum peut être conditionnée par trois contraintes [ROY 81] :

- *L'exclusivité mutuelle* qui présume que toutes les solutions potentielles sont mutuellement exclusives deux à deux. Cette condition permet d'assurer qu'une solution peut être choisie au détriment de toutes les autres solutions. Néanmoins, dans certains problèmes de rangement ou de tri, l'intérêt est plutôt porté sur la conservation des ensembles de solutions satisfaisantes non exclusives entre elles, sans toutefois exclure les autres. Dans de tels cas, les alternatives possibles ne peuvent être modélisées sous forme d'alternatives deux à deux mutuellement exclusives. C'est le cas par exemple lors de la sélection de candidats vérifiant un profil donné ou dans le secteur bancaire lors de l'attribution de crédits bancaires où les demandes de crédit ne sont pas deux à deux mutuellement exclusives. Ce genre de problèmes consiste donc à raisonner sur des alternatives compatibles pouvant être choisies simultanément et entre plutôt dans le cadre de l'aide à la décision[14].

- *L'exhaustivité des solutions explorées et considérées* : Le terme exhaustivité ne reflète pas ici la définition usuelle de recherche exhaustive souvent utilisée en optimisation et qui

[13] L'expression des critères en terme de dimensions a été traitée en détails dans [ROY 85].
[14] L'aide à la décision ne se limite pas aux problèmes violant les conditions d'optimisation. Ces conditions limitent seulement le champ d'application de l'optimisation. Les techniques d'optimisation peuvent toutefois être appliquées, pour modéliser ou résoudre un problème d'aide à la décision.

désigne la recherche de toutes les solutions possibles et donc l'exploration de tout l'espace de recherche. L'exhaustivité désigne ici l'ensemble des solutions relativement auxquelles l'optimum sera défini (l'espace de recherche exploré). Les solutions doivent être réparties dans l'espace de recherche exploré, de façon définitive, entre admissibles et inadmissibles ou prises en compte et ignorées. Les solutions non explorées ou non intéressantes (non admissibles) dans l'espace exploré sont considérées ignorées. La recherche de l'optimum s'effectue en comparant les solutions admissibles et prises en compte. Cet ensemble doit être clairement délimité et ne doit jamais être remis en cause lors du processus d'optimisation. L'introduction d'une nouvelle solution, ignorée précédemment, à cet ensemble implique nécessairement la perte de la validité de l'optimum trouvé.

- *La structure de pré-ordre complet* : Cette condition postule la possibilité de désigner de façon nette et précise une solution unique comme la meilleure parmi toutes les solutions potentielles. Cette condition peut être décomposée en deux sous-conditions :

  - *Le pré-ordre partiel* qui suppose que deux solutions sont nécessairement comparables. En effet, les évaluations des décideurs peuvent exprimer des relations binaires de préférence, d'indifférence ou d'incomparabilité entre les solutions. La condition de pré-ordre partiel exclut la relation d'incomparabilité. Deux types de relations sont donc possibles entre deux solutions : la préférence stricte (a est strictement préférée à b ou inversement) ou l'indifférence.

  - *La complète transitivité* : Les préférences des décideurs sur les solutions doivent être complètement transitives, i.e., a est meilleure que b, b est meilleure que c implique nécessairement a est meilleure que c [TVERSKY 69].

Cette 3ème condition n'est pas très contraignante dans le cas d'une optimisation mono-critère et mono-évaluation (en présence d'un seul décideur). Par contre, elle est rarement vérifiée en présence de critères multiples (la condition de pré-ordre partiel n'est pas vérifiée) et notamment en présence du facteur humain (la condition de complète transitivité est rarement respectée). Dans ce dernier cas, le décideur préfère a sur b, b sur c mais pas forcément a sur c. L'intransitivité de l'indifférence peut être illustrée par l'exemple suivant [SCHÄRLIG 85] : "Imaginez une série de sandwiches, formés de pain et de fromage, en proportions différentes : de plus en plus de pain et de moins en moins de fromage. En partant d'un premier sandwich, il est possible de substituer progressivement du pain au fromage, de sorte qu'à chaque substitution on soit indifférent entre ces deux termes, mais qu'on préfère le sandwich initial à celui finalement obtenu".

Il semble donc que l'optimisation trouve un sens plus évident dans les problèmes mono-critère ou les problèmes multicritères mono-évaluation (un seul acteur évalue les solutions) vérifiant la complète transitivité (le pré-ordre partiel). L'évaluation simultanée de plusieurs critères ne favorise pas la désignation de la meilleure solution face à une situation donnée. En plus la présence d'un ensemble de décideurs risque de désigner comme optimale une solution différente pour chaque point de vue (décideur) et qu'en fin de compte, aucun optimum commun à tous les points de vue ne puisse être dégagé.

I.1.4. Synthèse de l'optimisation

Le cadre de l'optimisation semble inadapté aux problèmes comportant des contraintes multiples et fortes ou qui ne vérifient pas les conditions d'optimisation. Il se limite souvent aux problèmes de grande taille ou peu-contraints (offrant un large choix de solutions). Cependant, les problèmes réels comportent couramment des contraintes importantes dont la violation risque de rendre la solution inapplicable dans la réalité et le processus de satisfaction est souvent plus important que la tâche d'optimisation elle-même. Les problèmes d'affectation sous contraintes semblent mieux gérer ce genre de situation et remédier à certaines insuffisances du cadre de l'optimisation.

I.2. L'affectation sous contraintes

Ce domaine s'intéresse aux problèmes faisant intervenir des contraintes fortes et multiples. Il est souvent utilisé pour chercher une solution, la meilleure solution ou l'ensemble des solutions satisfaisant toutes les contraintes, à l'aide des méthodes exactes qui représentent le cadre naturel pour la résolution de cette classe de problèmes.

Le cadre de problème de satisfaction de contraintes[15] (CSP) est généralement utilisé pour traiter les problèmes d'affectation sous contraintes. Nous présentons successivement les formalismes de CSP classique et d'optimisation sous contraintes, les limitations de ces formalismes vis-à-vis de certains types de problèmes (problèmes sur-contraints et sous-contraints) et les différentes extensions de CSP classique, à savoir les extensions simples et les méta-extensions.

[15] Notons que le cadre de CSP classique fait souvent référence aux CSPs avec des domaines finis (Finite constraint satisfaction problem) définis par [MACKWORTH 92] ou les problèmes d'étiquetage cohérent (consistent labelling problem) [TSANG 93].

I.2.1. CSP : Constraints Satisfaction Problem

Le formalisme CSP a été introduit dans les années 1970 pour modéliser un grand nombre de problèmes en intelligence artificielle et en recherche opérationnelle, tel est le cas par exemple, dans la vision où le but est de trouver une interprétation d'une scène conforme aux observations, dans l'ordonnancement et la planification de tâches, etc.

I.2.1.1. Définition

Un CSP est défini par un ensemble fini de variables, chacune pouvant prendre une valeur dans un domaine fini de valeurs possibles qui lui est associé [FREEMAN 90][FREUDER 89] [GU 89][JUSSIEN 97][MACKWORTH 85][MITTAL 87][RICCI 90] et [SHAPIRO 81]. Des contraintes portant sur des sous-ensembles de variables représentent des conditions à satisfaire et restreignent l'ensemble des valeurs pouvant être affectées simultanément à ces variables. Plus formellement, un CSP est défini par le triplet $<X,D,C>$:

- $X = \{x_1, x_2, ... x_i, ... x_n\}$ ensemble des n variables du problème;
- $D = \{\text{dom}(x_i) \mid x_i$ élément de X$\}$ ensemble des n domaines associés aux variables (i.e., leurs valeurs possibles);
- $C = \{c_1, ..., c_j, ... c_m\}$ ensemble des m contraintes, spécifiant les combinaisons de valeurs mutuellement compatibles.

a. Les variables

Les variables dans un CSP représentent des sous-problèmes à résoudre ou des décisions à prendre. Les domaines des variables permettent d'exprimer les alternatives envisageables et les contraintes entre variables reflètent les conditions à vérifier ou les instructions à suivre pour parcourir l'espace de recherche. Une variable est caractérisée par :

- *Identifiant* : nom de la variable.
- *Contraintes* : liste des contraintes (unaires, binaires, n-aires) applicables à la variable.
- *Variables liées* : liste des variables liées par une contrainte avec la variable concernée.
- *Domaine* : domaine de la variable, i.e., la liste des valeurs admissibles.

Le type des variables influe sur le choix de la méthode de résolution appropriée. Nous distinguons les variables numériques (définies sur des domaines contenant seulement des valeurs numériques), des variables booléennes (de domaines booléens) et les variables

symboliques (de domaines contenant seulement des types énumérés d'objets, e.g., petit, grand, …) [TSANG 93].

b. Les contraintes

[TSANG 93] définit une contrainte sur un ensemble de variables comme étant une restriction sur les valeurs qui peuvent être prises simultanément par ces variables. Une contrainte est définie par l'ensemble des variables sur lesquelles elle pèse (sa signature) et par l'ensemble des combinaisons de valeurs qu'elle autorise (liste de tuples de valeurs admissibles). Une contrainte est dite redondante si sa suppression ne change pas l'ensemble des solutions du problème[16]. Elle est satisfiable si elle possède une solution. Deux contraintes sont équivalentes si elles ont le même ensemble de solutions. Plus formellement, une contrainte c_j se caractérise par :

- *Var* $(c_j) = \{x_{j1}, ..., x_{jn}\}$: l'ensemble des n variables liées par c_j ; n est appelée arité de la contrainte c_j (le nombre de variables qu'elle implique). Les variables sur lesquelles porte une contrainte sont dites liées (par cette contrainte). Une contrainte unaire agit sur la valeur d'une seule variable. Une contrainte binaire agit sur la combinaison des valeurs d'un couple de variables. Une contrainte multiple (n-aire) quant à elle concerne la combinaison des valeurs de plus de deux variables.

- *Cond* (c_j) : la condition associée à la contrainte c_j. Elle spécifie les combinaisons de valeurs compatibles pour les variables liées par c_j (les n-uplets autorisés par cette contrainte). La condition de la contrainte représente un sous-ensemble du produit cartésien des domaines des variables liées par cette contrainte : dom $(x_{j1}) \times ... \times$ dom (x_{jn}).

En pratique, les contraintes peuvent être définies en extension (de manière explicite) en spécifiant explicitement l'ensemble des tuples autorisés ou interdits ou en intension/compréhension (de manière implicite) par la fonction représentative de l'ensemble des tuples autorisés (sous forme d'équations, inéquations, matrices, etc.).

I.2.1.2. Solution à un CSP

Une solution (affectation) potentielle du problème consiste à affecter à chaque variable une valeur de son domaine. L'ensemble des solutions potentielles est donc représenté par le produit cartésien des domaines des variables. Une évaluation (ou une instanciation) d'un CSP

[16] Dans les problèmes complexes, les contraintes redondantes sont généralement difficiles à détecter.

est une affectation de valeurs aux variables. Une solution est une évaluation qui satisfait toutes les contraintes. Une instanciation est dite consistante si elle satisfait toutes les contraintes portant sur ses variables. Une solution d'un CSP peut donc être vue comme une instanciation consistante de toutes les variables du problème. Une instanciation d'un ensemble de variables $V \subseteq X$ est dite partielle si $V \neq X$ et complète sinon ($V = X$). Un CSP est satisfiable si une solution existe pour le résoudre. Deux CSPs sont équivalents s'ils ont le même ensemble de variables et les mêmes tuples de solutions. Un algorithme est dit solide si toutes les solutions qu'il fournit satisfont toutes les contraintes.

I.2.1.3. Représentation des CSPs

La structure de graphe de contraintes (appelé parfois réseau de contraintes) constitue un cadre naturel pour représenter les problèmes faisant intervenir des contraintes. Un graphe est constitué par un ensemble de nœuds et d'arcs. Les nœuds représentent les étapes à parcourir (problèmes à résoudre) et les arcs expriment les règles de passage entre ces nœuds (l'ordre de résolution des problèmes). Plus formellement, un graphe est un tuple <N,A> où N est un ensemble de nœuds et $A \subseteq N \times N$ est un ensemble d'arcs reliant les nœuds.

- *Les nœuds* : Une structure nœud (node) est souvent associée à une variable du problème. Elle peut aussi représenter une contrainte à satisfaire (une instanciation de variables), un niveau de préférence (cas des extensions de CSP), un attribut ou entité (cas d'un modèle conceptuel), un agent (cas de CSP distribué), etc.

- *Les arcs* : Très souvent, une structure arc modélise une condition ou une sous-condition associée à une contrainte. Une contrainte unaire est représentée par une boucle, i.e., un arc partant et se terminant au même nœud et peut être immédiatement satisfaite par la réduction du domaine de la variable concernée (par consistance de nœud). Les arcs représentant les contraintes unaires peuvent être supprimés pour simplifier les algorithmes de recherche et obtenir ainsi un réseau de contraintes[17].

La façon la plus simple pour représenter un CSP binaire, consiste à exprimer une variable par un nœud et une contrainte par un arc (figure 7). Un arc représente donc une ou plusieurs contraintes entre deux variables (nœuds) aux 2 extrémités de l'arc. Cette représentation se limite au cadre de CSP binaire, puisque dans ce cas, un arc ne peut lier que deux nœuds.

[17] Un réseau de contraintes est un graphe connecté et sans boucles.

D'autres représentations sont envisageables et un arc peut représenter un lien quelconque entre deux nœuds (figure 8).

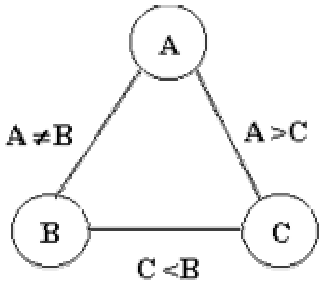


Figure 7 : Exemple de graphe représentant 3 variables (A, B, C) et 3 contraintes (A>C, A ≠B, C<B) liant ces variables.

Application	Nœud	Arc
CSP binaire classique	Une variable du problème.	Une contrainte.
CSP n-aire classique	Instanciation de variables. Une contrainte.	Appartenance de 2 variables à la même contrainte. Partage d'une variable entre 2 contraintes.
Extensions de CSP	Un niveau de préférence.	Lien de coordination. Passage des contraintes les + prioritaires à celles - prioritaires.
Méthodes de voisinage	Une configuration.	Lien de voisinage
Problèmes distribués	Un agent. Une hiérarchie d'agents.	Echanges de données entre des agents. Appartenance à la même hiérarchie.

Figure 8 : Exemples (non exhaustifs) de codification de problèmes à l'aide de graphe de contraintes.

En effet, bien que la plupart des travaux se limitent aux CSPs binaires où les contraintes relient 2 variables au maximum, la notion de graphe reste un moyen générique de représentation applicable à de nombreuses situations. Plusieurs travaux ont montré qu'il est possible de convertir un CSP n-aire (contraintes reliant jusqu'à n variables) en un CSP binaire équivalent [BACCHUS 98][BESSIERE 99][STERGIOU 99][SMITH 00]. Cette possibilité d'exprimer des contraintes d'arité supérieure à 2 en terme de contraintes binaires est importante du point de vue théorique puisqu'elle permet de se restreindre au cas de CSPs binaires (les CSPs binaires deviennent représentatifs de tous les CSPs). La conversion d'un CSP n-aire à un CSP binaire équivalent est basée sur l'introduction de nouvelles variables qui

encapsulent les variables liées par des contraintes. Les nouvelles variables introduites (variables encapsulantes) ont pour domaine le produit cartésien des domaines des variables individuelles encapsulées.

Cependant, même si théoriquement tout problème non binaire peut être codé sous une forme binaire, des considérations pratiques rendent inexploitables certaines de ces transformations. La binarisation de contraintes est donc un outil principalement théorique qui sert à justifier les travaux et les algorithmes qui se limitent aux contraintes binaires ou unaires et la plupart des applications pratiques travaillent ainsi avec les contraintes n-aires directement [BACCHUS 98][BESSIERE 99].

I.2.2. CSOP : Les problèmes d'optimisation sous contraintes

Selon les besoins de l'application, la résolution d'un CSP consiste à trouver une solution (e.g., analyse des scènes dans la vision), l'ensemble de solutions (dans certains cas de planification ou de programmation logique) ou la meilleure solution. Il s'agit dans ce dernier cas de problème mixte de satisfaction et d'optimisation.
Le formalisme CSP est considéré comme le cadre le plus naturel pour coder les problèmes faisant intervenir des contraintes. Cependant, de plus en plus de problèmes s'avèrent mixtes de satisfaction de contraintes et d'optimisation. D'où l'extension du formalisme CSP classique au formalisme d'optimisation sous contraintes (CSOP). Le cadre de CSOP sert à traiter les problèmes d'optimisation faisant intervenir des contraintes fortes et peut donc être considéré comme une extension de CSP classique avec une fonction à optimiser relative à une évaluation numérique de chaque solution. Plus formellement un CSOP est défini par un quadruplet $<X,D,C,F>$ où :

- $<X,D,C>$ est un CSP classique.

- F : La fonction à optimiser. Chaque solution s de S est évaluée par la fonction F(s) et l'objectif consiste à trouver la solution optimale s* au sens de la fonction F de coût ou de profit.

La façon la plus naïve pour résoudre un CSOP consiste à trouver toutes les solutions possibles (cas de CSP) afin de choisir la meilleure. Une solution plus efficace consiste à utiliser des heuristiques au cours de la recherche afin d'éliminer les parties de l'espace qui n'ont aucune chance de contenir une solution meilleure que celles trouvées auparavant, i.e., soit qu'il n'y a pas de solutions, soit que les solutions sont sous-optimales (cf. méthode

Branch and Bound [LAWLER 96]).
Cependant, les problèmes concrets sont en général beaucoup plus complexes que les cas théoriques à cause de la complexité du codage des informations spécifiques à la problématique et l'imprécision, dans certains cas, des données manipulées. Certains problèmes ne peuvent donc être exprimés à l'aide de CSOP ou de CSP et nécessitent le recours à d'autres extensions pour faire face aux limites de ces formalismes classiques.

I.2.3. Limites des formalismes classiques de CSP et CSOP

L'utilisation des formalismes de CSP et CSOP dans le cadre d'applications réelles met en évidence un ensemble de limitations. Ces limitations ne sont pas nécessairement liées à l'inefficacité de ces formalismes mais plutôt à leur simplicité qui ne permet pas de construire une modélisation réaliste assurant la prise en compte des notions de préférences [FARGIER 93a] et d'incertitude [FARGIER 93b].

I.2.3.1. Les préférences

Un CSP classique a pour finalité l'expression et la résolution des problèmes faisant intervenir des contraintes[18]. Trouver une solution à un CSP consiste à affecter à chaque variable une valeur de son domaine de manière à ce que toutes les contraintes soient respectées. L'ensemble des instanciations est réparti en deux classes : celles qui sont effectivement des solutions du problème (affectations consistantes) et celles qui ne le sont pas (affectations inconsistantes). L'introduction de préférences permet de raffiner cet ordre en créant des classes d'instanciations intermédiaires. La notion de préférences peut également s'appliquer sur les contraintes afin de distinguer celles qu'il faut absolument satisfaire (contraintes dures) de celles qu'il est préférable de satisfaire (contraintes molles ou souples) [TANGUIANE 91][VINCKE 82]. La prise en compte des préférences permet de guider la résolution des problèmes sous-contraints (under-constrained problems) et sur-contraints (over-constrained problems).

a. Les problèmes sous-contraints

Certains problèmes ne sont pas assez contraints et offrent un large choix de solutions. Il est donc nécessaire de donner un supplément d'information pour orienter le choix d'une

[18] Les contraintes dans un CSP classique ont le même niveau d'importance (préférence).

solution. Ces informations peuvent être assimilées à des préférences du système et permettent ainsi de guider la résolution du problème. Nous distinguons trois formes de préférences:

- Un complément d'information sur les données d'entrée, avec par exemple, l'ajout de contraintes afin de restreindre le nombre de solutions possibles.

- Un contrôle sur le mécanisme de résolution, avec par exemple la définition d'heuristiques pour guider la recherche de solution.

- Un critère de sélection appliqué sur les solutions produites (ou de priorité sur les contraintes) afin de choisir les plus pertinentes.

b. Les problèmes sur-contraints

Certains problèmes sont trop contraints, à savoir que leur système de contraintes associé ne permet pas de trouver une évaluation qui satisfait toutes les contraintes. Deux stratégies différentes permettent de résoudre les CSPs sur-contraints :

- Appliquer un ordre de préférence sur les contraintes : La prise en compte des préférences permet de relâcher les contraintes les moins prioritaires et de trouver ainsi les solutions qui sont jugées les moins mauvaises (satisfaction des contraintes les plus prioritaires).

- Considérer le même niveau de préférence sur les contraintes et chercher les solutions qui satisfont le maximum de ces contraintes.

I.2.3.2. L'incertitude

Dans de nombreux cas pratiques, la description du problème n'est pas assez détaillée (ou incomplète) et/ou certaines données sont incertaines ou vagues. C'est le cas par exemple, dans la gestion à long terme des successions culturales où la modification de la situation du marché peut influer sur les décisions prises ou encore lors de la prévision de la qualité du sol et son comportement vis-à-vis de l'absorption de l'eau où les changements climatiques peuvent modifier les données initiales.
Pour représenter ce manque d'information ou l'incertitude [FARGIER 93b], il est possible d'associer aux données du problème des mesures d'incertitude, sous forme de probabilités quand la nature de ces données l'autorise (quand l'incertitude est de type hasard) ou de possibilités/nécessités, qui permettent une mesure plus qualitative, lorsque l'incertitude est plutôt de type ignorance (sans interprétation statistique).

Afin de remédier aux problèmes issus des situations d'incertitudes ou de préférences, certaines extensions au cadre du CSP ont été proposées comme des alternatives permettant de traiter de tels problèmes en se basant généralement sur la relaxation de contraintes.

I.2.4. Les extensions du CSP classique

L'expression et le traitement de préférences ou d'incertitude dans les problèmes de satisfaction de contraintes a fait l'objet de diverses extensions du cadre CSP classique afin de permettre l'introduction de ces paramètres[19] à l'aide d'un critère de distance (CSPs partiels [FREUDER 92]), de coût (CSPs pondérés, CSPs à pénalité [FREUDER 92]), de nécessité/priorité (CSPs possibilistes [SCHIEX 92], approche lexicographique [FARGIER 93a]), d'appartenance (CSPs flous [ROSENFELD 76]), de probabilité/certitude (CSPs probabilistes [FARGIER 93b]), de valuation (CSPs valués [SCHIEX 95]) ou de force/poids (hiérarchies de contraintes [BORNING 87]). Ces différents critères servent à évaluer les solutions potentielles via des opérateurs spécifiques à chaque formalisme d'extension, comme par exemple :

- L'opérateur Max pour les CSPs possibilistes et flous.
- L'opérateur d'Addition utilisé dans le cadre des CSPs à pénalité et CSPs pondérés.
- L'opérateur d'Union utilisé dans le cadre de l'approche lexicographique.
- L'opérateur de Multiplication pour les CSPs probabilistes.

Nous distinguons les extensions simples traitant un type particulier de préférences et les extensions plus génériques ou méta-extensions applicables à des problèmes divers et capables d'exprimer les différents types d'extensions simples.

I.2.4.1. Les extensions simples

Elles concernent des cadres basiques sur des types particuliers de préférences. Nous citons :

- *Les CSPs flous ou Max-Min (FCSPs)* : Ils ont été introduits pour la première fois dans [ROSENFELD 76]. A la différence des CSPs classiques où les contraintes sont soit satisfaites soit insatisfaites, les contraintes dans les FCSPs sont plutôt satisfaites à un certain degré. Chaque contrainte est ainsi définie non plus par les combinaisons de valeurs qui la satisfont,

[19] L'introduction de ces paramètres se fait au niveau des contraintes ou des tuples de valeurs affectées aux variables.

mais par un ensemble flou correspondant à l'ensemble des valeurs qui la satisfont plus ou moins.

- *Les CSPs pondérés (WCSPs)* : Les WCSPs sont généralement utilisés pour modéliser les problèmes d'optimisation où l'objectif est de minimiser le coût total en nombre de ressources, en temps, en espace, etc. A chaque tuple de valeurs est attribué un coût associé. La fonction de coût est ainsi définie en additionnant les coûts de toutes les contraintes (en considérant le coût du tuple choisi pour chaque contrainte). Le but consiste à trouver les tuples qui minimisent le coût total de la solution proposée.

- *Les CSPs probabilistes (Prob-CSPs)* : Le formalisme des réseaux de contraintes probabilistes [FARGIER 93b] permet d'exprimer des problèmes partiellement connus ou des problèmes de décision dans l'incertain où l'existence d'un sous-ensemble de contraintes dans le problème réel est incertaine. A chaque contrainte est attribuée une probabilité d'existence (entre 0 et 1) et la solution optimale représente une instanciation qui maximise sa probabilité d'être une solution du problème final. L'objectif consiste donc à trouver une affectation de toutes les variables qui maximise la probabilité d'être une solution dans le monde réel.

- *Les CSPs possibilistes* : Définis dans [SCHIEX 92], les CSPs possibilistes permettent d'exprimer des nécessités ou des priorités entre contraintes. Chaque contrainte est annotée avec une priorité (réel) entre 0 et 1. Plus la priorité d'une contrainte est grande, plus il est nécessaire de la satisfaire. Une contrainte de priorité 1 doit nécessairement être satisfaite. Le jugement de la qualité d'une solution se fait seulement en fonction de la contrainte la plus importante (ou prioritaire) violée et l'objectif consiste généralement à rechercher une instanciation qui minimise la priorité la plus élevée des contraintes violées.

- *L'approche lexicographique* : Il s'agit d'une amélioration de l'approche possibiliste en prenant en compte le nombre de contraintes violées à chacun des niveaux d'importance et non seulement au niveau le plus important. La qualité d'une instanciation n'est donc pas jugée uniquement en fonction de l'évaluation de la contrainte violée la plus importante mais plutôt en fonction des évaluations de toutes les contraintes violées [FARGIER 93a].

- *Les CSPs à pénalité* : Ils ont été introduits en tant que CSPs partiels dans [FREUDER 92]. A chaque contrainte est attribuée une valeur de pénalité dans le but de minimiser le nombre de contraintes violées ou une somme pondérée (e.g., la somme des pénalités de la solution finale). L'approche similaire en logique propositionnelle définit le problème MAX-SAT [GAREY 79].

I.2.4.2. Les extensions génériques ou méta-extensions

Certaines extensions génériques ont été proposées pour représenter et exprimer les différentes extensions de CSP classique. Leur point commun consiste à associer une évaluation à chaque contrainte ou à chaque tuple de valeurs des variables liées. L'ensemble des évaluations est intégré dans une fonction générale d'optimisation basée sur des opérateurs monotones ou idempotents dépendants de l'approche. Les différentes extensions simples peuvent alors être obtenues par la spécification de la structure générale des méta-extensions.

- *Les CSPs partiels (PCSPs)* : Le formalisme PCSP [FREUDER 89][FREUDER 92] est une extension de CSP classique, qui autorise la relaxation et l'optimisation des problèmes via l'affaiblissement des problèmes sur-contraints qui n'ont aucune solution ou pour lesquels la recherche de solution nécessite beaucoup de temps. L'objectif est de trouver une assignation qui satisfait un nombre maximum de contraintes. Un PCSP implique la recherche de valeurs pour un sous-ensemble de variables qui satisfont un sous-ensemble des contraintes. Le CSP originel est transformé à un problème plus faible (moins contraint) et plus facile à résoudre. Un CSP peut être affaibli de différentes façons :

  - Elargir le domaine d'une ou de plusieurs variables ou contraintes afin d'autoriser des combinaisons de valeurs acceptables supplémentaires.

  - Supprimer une ou plusieurs variables ou contraintes.

Contrairement aux CSOPs qui exigent que toutes les contraintes soient satisfaites, un PCSP consiste à trouver une solution qui minimise le nombre de contraintes violées (ou maximise le nombre de contraintes satisfaites). Les CSOPs peuvent donc être vus comme un cas particulier de PCSPs où toutes les contraintes sont satisfaites. De la même manière, les PCSPs représentent un cas particulier de CSOP où la fonction à optimiser est le nombre de contraintes satisfaites.

- *Les hiérarchies des contraintes* : Les hiérarchies des contraintes [BORNING 87] sont introduites pour décrire les CSPs sur-contraints en spécifiant les contraintes avec un coefficient hiérarchique (un poids) de préférence afin de distinguer les contraintes faibles des contraintes fortes. Lorsque deux contraintes en opposition ne peuvent être satisfaites simultanément, la satisfaction de celle de poids le plus fort est favorisée. Cette approche interdit donc aux contraintes les plus faibles d'influencer le résultat au détriment des contraintes les plus fortes. Nous distinguons deux stratégies de résolution différentes :

- La méthode affinée (refining algorithms) : Elle consiste à choisir l'ordre de contraintes à prendre en compte afin de satisfaire en priorité celles du niveau le plus fort en passant aux niveaux inférieurs.

- La propagation locale (local propagation algorithms) : C'est une méthode plus avantageuse et plus simple qui consiste à résoudre progressivement les hiérarchies de contraintes en sélectionnant à plusieurs reprises les contraintes satisfiables uniquement. Dans cette technique, une seule contrainte est prise en compte (à la fois) pour déterminer la valeur d'une variable. Une fois que cette valeur est déterminée, le système considère une autre contrainte pour affecter la variable suivante, et ainsi de suite.

- *Les CSPs Valués (VCSPs)* : Ils représentent un cadre général proposé dans [SCHIEX 95][SCHIEX 97] pour exprimer les différentes extensions du formalisme de CSP classique. Une valuation exprimant une probabilité d'existence ou d'interdiction est associée à chaque contrainte ou à chaque n-uplet définissant la contrainte. Ces valuations expriment l'impact de la violation d'une contrainte ou du choix de l'affectation des variables sur la qualité de la solution. Si plusieurs contraintes sont violées, la combinaison des valuations qui leur sont associées via une loi de composition interne définit l'impact cumulé de ces violations.

- *Les CSPs Mixtes* : Ce cadre distingue les variables contrôlables par l'utilisateur (variables de décision) et les variables incontrôlables par l'utilisateur (variables d'état ou d'environnement). L'objectif est de trouver une affectation des variables contrôlables telle que toutes ses extensions possibles aux variables incontrôlables soient des solutions. Si une distribution de probabilité est disponible sur les domaines de toutes les variables incontrôlables, l'objectif peut consister à trouver une affectation des variables contrôlables qui maximise sa probabilité d'être solution au problème [FARGIER 96].

D'autres extensions existent comme les Semiring-based Constraint Satisfaction [BISTARELLI 97][BISTARELLI 99] ou les CSPs temporels qui s'intéressent à la résolution des problèmes de satisfaction de contraintes temporelles où les variables représentent le temps et les contraintes les relations temporelles entre elles.

I.2.5. Conclusion : Quel formalisme utiliser ?

Le choix du formalisme de codification d'un problème d'affectation sous contraintes dépend des spécificités de la problématique et du point de vue du concepteur. En effet, le cadre de CSP est souvent utilisé pour chercher une solution ou l'ensemble des solutions d'un

problème, à l'aide d'une méthode exacte. La branche de CSOP est utilisée, tout comme la recherche opérationnelle, pour chercher la meilleure solution, i.e., trouver une solution qui satisfait toutes les contraintes et qui optimise une fonction d'évaluation. Elle peut aussi être utilisée pour chercher une solution de bonne qualité qui satisfait l'ensemble des contraintes. Les problèmes de CSOP peuvent être résolus à l'aide de méthodes exactes (si la satisfaction de toutes les contraintes est primordiale et si la contrainte de temps est absente) ou à l'aide de méthodes approchées pour assurer un rapport qualité/coût.
Quand le problème est sur-contraint, c'est à dire qu'aucune solution satisfaisant toutes les contraintes ne peut être trouvée, les extensions de CSP sont utilisées pour satisfaire le maximum de contraintes (si toutes les contraintes ont la même préférence) ou celles les plus pertinentes à l'application.

I.2.6. Synthèse de l'affectation sous contraintes

Le formalisme d'affectation sous contraintes, mieux adapté aux problèmes sur-contraints n'accorde pas suffisamment d'intérêt à la modélisation des contraintes et sa première préoccupation consiste à choisir ou développer la bonne technique permettant de satisfaire ces contraintes. L'analyste prend souvent en charge, lui-même, la résolution des problèmes liés à l'aspect sur-contraint ou imprécis du problème, sans faire appel à l'aide des décideurs. Ces derniers restent les acteurs les plus concernés par la relaxation de leurs contraintes et leur seule implication ressentie dans certains travaux, se limite à l'expression de leurs préférences ou leurs points de vue vis-à-vis d'une situation donnée. L'aide à la décision se propose de combler ce manque d'intégration des décideurs dans les processus de prise de décision.

I.3. L'aide à la décision

L'aide à la décision consiste à assister les décideurs et les aider à mieux exprimer leurs choix et préférences vis-à-vis d'une situation donnée. B. Roy [ROY 90] définit l'aide à la décision comme étant « l'activité de celui qui, prenant appui sur des modèles clairement explicités mais non nécessairement complètement formalisés, aide à obtenir des éléments de réponse aux questions que se pose un intervenant dans un processus de décision ». En effet, contrairement à l'optimum, la bonne décision ne se découvre pas, mais se construit. Selon B. Roy « La bonne décision est un construit qui s'élabore en relation avec des finalités, avec un contexte, avec des capacités d'appropriation par le milieu, … c'est pourquoi la démarche à suivre n'est pas une démarche de découverte mais une démarche de construction »

[COURBON 93].

L'aide à la décision trouve un sens plus évident dans les problèmes ne vérifiant pas les conditions d'optimalité ou nécessitant une intervention permanente des décideurs dans le processus de décision et face auxquels l'optimisation se trouve impuissante.
Cependant, la tâche d'optimisation n'est pas tout à fait écartée dans les démarches d'aide à la décision et peut servir en tant que technique de progression vers l'obtention d'éléments de réponse ou la construction de la décision. B. Roy précise dans ce cadre que « Les calculs d'optimisation restent fondamentaux pour éclairer des décisions », « La démarche d'optimisation est très efficace pour apporter des éléments d'analyse et de réponse mais pas pour dicter la décision ou pour prétendre découvrir un optimum réel » [COURBON 93].

Les méthodes employées appartiennent initialement à la recherche opérationnelle dont l'objectif consistait, à optimiser le fonctionnement de systèmes connus et maîtrisés. Ces techniques montrent leurs limites dans les problèmes renfermant des objectifs à aspects multiples tels qu'économique (optimisation de profit, de coût), social (appartenance, impact sur la société), moral (satisfaction morale), etc.
Selon B. ROY, « Beaucoup d'échecs de la recherche opérationnelle s'expliquent par l'acharnement mis à ne rien vouloir faire d'autre que d'apporter la solution à un problème considéré comme définitivement bien posé alors qu'il aurait été beaucoup plus efficace d'utiliser les modèles et procédures préconisés comme leviers pour faire évoluer la formulation du problème » [COURBON 93].
Différentes méthodes d'aide à la décision sont conçues pour étendre les méthodes classiques et affronter ce type de problématiques, en l'occurrence les méthodes multicritères [ROY 85].

La partie suivante sera consacrée à une courte présentation des démarches d'aide à la décision, suivie d'une description des termes permettant de caractériser un problème de décision avant de synthétiser ce cadre de représentation des problèmes.

I.3.1. Les démarches d'aide à la décision

Les travaux qui traitent des problèmes de prise de décision s'inscrivent généralement dans l'une des démarches suivantes[20] [BELL 88] : La démarche descriptive, prescriptive ou normative.

[20] B. Roy [ROY 85] distingue l'approche descriptive et l'approche constructive.

La démarche *descriptive* [BOUYSSOU 84] a pour but la description et la prévision du comportement du décideur. Elle suppose qu'il préexiste à toute analyse du problème, une vérité cachée. Le rôle des modèles descriptifs est de dévoiler cette vérité [ROY 85].

La démarche *prescriptive* s'intéresse aux conseils et instructions qui peuvent être fournis au décideur afin qu'il améliore ses décisions. Ces instructions doivent être en accord avec les besoins et les capacités cognitives des individus pour lesquels elles sont conçues. Les modèles prescritifs sont évalués par leur valeur pragmatique, i.e., leur capacité d'aider le décideur à améliorer ses décisions [BELL 88].

La démarche *normative* consiste à définir des principes et des règles qui traduisent un comportement logique et rationnel. Cette démarche attribue à ces règles la valeur d'une vérité indiscutable ou de règle idéale que le décideur doit rationnellement suivre [ROY 90]. Ces règles sont ainsi perçues comme des hypothèses de travail pour guider l'interaction entre l'homme d'étude et le décideur.

I.3.2. Description d'un problème d'aide à la décision

Les termes essentiels qui permettent de décrire un problème d'aide à la décision sont les alternatives et les acteurs.

a. Les alternatives

L'ensemble des alternatives désigne des possibilités, des variantes, des scénarios, des configurations, des solutions ou des actions possibles, sur lesquels porte la décision. L'identification de cet ensemble est une tâche primordiale dans la définition du problème. Dans notre cas pratique, les alternatives représentent l'ensemble des solutions ou configurations (affectation des cultures sur les parcelles) satisfaisant les contraintes du système. Le grand nombre d'alternatives ne pouvant être défini explicitement, un ensemble de contraintes détermine les alternatives possibles caractérisées par les valeurs (cultures) que peuvent prendre les variables du système (parcelles).

b. Les acteurs

B. Roy [ROY 85] définit un acteur comme étant un individu ou un groupe d'individus qui, par son système de valeurs, que ce soit du fait de ses intentions ou par la manière dont il fait intervenir celles d'autres individus, influence directement ou indirectement la décision. Un acteur est donc un individu intervenant directement ou indirectement dans la prise de

décision. Son rôle est susceptible d'évoluer tout au long du processus de décision. Pour qu'un groupe d'individus soit identifié comme un seul et même acteur, il faut que « relativement au processus, les systèmes de valeurs, systèmes informationnels et réseaux relationnels des divers membres du groupe n'aient pas à être différenciés » [ROY 96]. En général, deux types d'acteurs peuvent être distingués : Le décideur et l'analyste.

- *Le décideur* : Un décideur est un individu (ou un groupe d'individus), à qui sont destinées les solutions proposées et à qui revient souvent la prise de décision dans l'environnement de la problématique. Le décideur (ou encore l'utilisateur) a pour responsabilité l'expression de ses besoins et l'évaluation des différentes alternatives possibles. Trois types de décideurs peuvent être distingués dans notre problématique : Les agriculteurs, les agronomes et les instances locales. Suivant le nombre de décideurs, deux situations sont distinguées :

    - Situation de mono-décideur (mono-évaluation) : Une seule partie prenante influence le processus de décision. L'évaluation des solutions se fait ainsi selon un seul point de vue.

    - Situation de multi-décideurs (multi-évaluations) : Plusieurs parties prenantes influencent la prise de décision. Chaque solution peut être évaluée de manière objective (commune à l'ensemble des décideurs) ou subjective (évaluation particulière pour chaque point de vue). Une tâche délicate consiste à adapter la démarche de décision à la dimension multi-décideurs afin d'étudier la divergence entre les points de vue et de prévoir les éventuels conflits entre eux.

- *L'analyste ou l'homme d'étude* : L'analyste a pour tâche de modéliser le problème de décision et de mener à bien les différentes phases de résolution du problème à savoir le suivi de la négociation, la présentation des résultats au décideur etc. A. Tecle [TECLE 91] définit l'activité de l'analyste comme étant la formulation et l'analyse qualitative et quantitative du problème.

En se référant à [MAYSTRE 94], F. Joerin [JOERIN 97] définit l'analyste comme étant l'individu ou le groupe d'individus, qui prend en charge l'aide à la décision en utilisant des modèles plus ou moins formalisés et dont la tâche consiste à accompagner les acteurs dans la démarche d'aide à la décision. Nous utilisons dans ce rapport indifféremment les termes d'analyste, concepteur ou homme d'étude.

L'analyste dans notre cas représente l'informaticien chargé du développement de l'application. Afin de mener à bien le processus de résolution, l'homme d'étude est aidé par des experts géographes et agronomes qui ont pour tâche d'extraire les connaissances du

terrain et de mener des enquêtes exhaustives auprès des agriculteurs et des instances locales ainsi que la contribution à l'évaluation des solutions proposées.
L'homme d'étude doit veiller à assurer une certaine neutralité afin de ne pas influencer les choix des décideurs. Le processus de l'aide à la décision dépend en partie de l'homme d'étude et de son impact sur la prise de décision. Le résultat d'un processus de décision pourrait ainsi changer en changeant l'homme d'étude.

I.3.3. Synthèse de l'aide à la décision

Le processus d'aide à la décision consiste à aider le décideur à prendre une décision vérifiant des contraintes et répondant à des conditions prédéfinies. Cependant, dans cette communauté, l'aspect contraintes est souvent peu évoqué voire absent dans les démarches de décision et semble être la dernière préoccupation de l'homme d'étude. L'intérêt se porte essentiellement sur l'interaction et l'intégration du décideur dans la prise de décision. En effet, nous pensons que l'aspect contraintes, qui restreint l'espace de recherche et contraint la qualité de la décision choisie, ne puisse être écarté de toute procédure de prise de décision.

I.4. Synthèse des cadres formels

L'étude des domaines de l'optimisation, l'affectation sous contraintes et l'aide à la décision permet de constater que ces trois disciplines, sont plutôt complémentaires que concurrentes d'autant plus qu'elles s'attaquent aux mêmes catégories de problèmes. Les frontières entre ces disciplines restent arbitraires et mal définies à cause du manque de connaissances des chercheurs vis-à-vis des autres disciplines. Les techniques utilisées par ces formalismes sont souvent issues des mêmes principes malgré une divergence des appellations et des termes utilisés. D. Dubois remarque à ce propos : « Je pense qu'il y a actuellement une super-spécialisation des disciplines qui fait que chacun reste dans son coin, sa petite chapelle bien que l'on retrouve les mêmes outils algorithmiques partout… mais cette super-spécialisation est tout de même quelque peu artificielle » « On a des fois l'impression qu'on redécouvre les mêmes choses » [COURBON 93].

La partie suivante sera consacrée à une présentation des différentes techniques de résolution utilisées pour affronter les problèmes s'inscrivant dans le cadre de l'une de ces trois disciplines.

II. Méthodes de résolution des problèmes

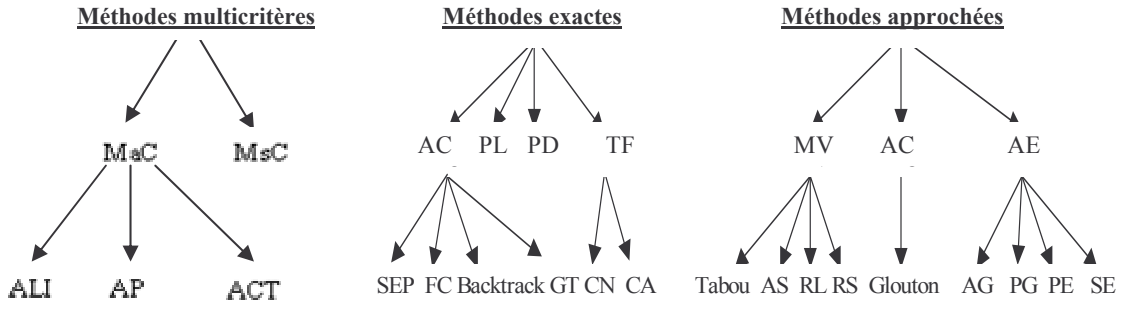
De nombreuses méthodes de résolution ont été développées en recherche opérationnelle (méthodes de voisinage et certaines approches de construction) et en intelligence artificielle (les algorithmes évolutifs, les méthodes de réparation et les techniques à base de backtrack) pour être exploitées dans la résolution des problèmes. Ces méthodes peuvent être réparties en trois grandes classes (figure 9) :

- *Les méthodes exactes ou complètes* : Leur objectif essentiel est de garantir la complétude de la résolution, i.e., toutes les solutions du problème peuvent être trouvées.

- *Les méthodes approchées* qui perdent la complétude pour gagner en efficacité. Leur objectif premier consiste à trouver une solution de bonne qualité en un temps de calcul raisonnable (rapport temps/qualité) sans toutefois garantir une solution optimale (elles convergent généralement vers les optimums locaux).

- *Les méthodes multicritères* : Certains problèmes comportent plusieurs critères à améliorer simultanément. Les méthodes multicritères ont généralement pour objectif la recherche d'un bon compromis entre les différents critères.

Nous présentons brièvement ces différentes techniques de résolution. Cependant, ce document n'est pas le lieu d'un exposé complet de ces différentes approches. Nous nous contentons donc d'un aperçu sommaire en renvoyant le lecteur intéressé à la bibliographie.



AC : Approche de Construction
PL : Programmation Linéaire
PD : Programmation Dynamique
TF : Techniques de Filtrage
MV : Méthodes de Voisinage
AE : Algorithmes Evolutifs
SEP : Séparation et Evaluation Progressive
FC: Forward Checking
GT : Générate and Test
MaC : Méthodes Avec Compensation
ACT : Agrégation Complète Transitive
ALI : Agrégation Locale Itérative

CN : Consistance des Nœuds
CA : Consistance des Arcs
AS : Algorithmes de Seuil
RL :Recherche Locale
RS : Recuit Simulé
AG : Algorithmes Génétiques
PG : Programmation Génétique
PE : Programmation Evolutive
SE : Stratégies Evolutives
MsC : Méthodes Sans Compensation
AP : Agrégation Partielle

Figure 9 : Classification des méthodes de résolution approchées, exactes et multicritères.

II.1. Les méthodes exactes

Le principe des méthodes exactes consiste à rechercher, souvent de manière implicite, une solution, la meilleure solution ou l'ensemble des solutions d'un problème. Elles sont généralement basées sur des heuristiques spécifiques pour guider la recherche de solution. Le temps de calcul nécessaire augmente en général exponentiellement avec la taille[21] du problème à résoudre ce qui explique la limite de ces méthodes face aux applications de taille importante. Les méthodes exactes sont principalement utilisées dans l'un des cas suivants :

- Quand l'optimalité est primordiale.
- Quand l'ensemble des solutions satisfaisant les contraintes doit être trouvé.
- Quand la taille du problème est raisonnable ou en absence de contrainte de temps.

Parmi les méthodes exactes, nous trouvons la plupart des méthodes traditionnelles de programmation sous contraintes, à savoir, la programmation linéaire, la programmation dynamique, certaines approches de construction et les techniques de pré-traitement.

II.1.1. La programmation linéaire

La programmation linéaire [BERTSIMAS 97], branche de la programmation mathématique, est une méthode de résolution de problèmes visant à rendre optimale une fonction linéaire, appelée fonction objectif, en fonction de contraintes. Ces contraintes sont généralement représentées par des systèmes d'équations ou d'inéquations linéaires et le mécanisme de résolution cherche à trouver un vecteur vérifiant le système de contraintes et optimisant la fonction objectif. Nous distinguons la programmation linéaire simple où toutes les variables sont réelles et la programmation linéaire en nombres entiers manipulant des variables entières.

Un programme linéaire est un problème d'optimisation où le but est de maximiser ou de minimiser une fonction objectif exprimée en fonction des variables du système (appelées aussi variables de décision). La programmation linéaire est utilisée dans le cas où les variables sont numériques et toutes les contraintes sont des équations ou inéquations linéaires (e.g., l'algorithme du simplexe [PAPARRIZOS 03]).

[21] La taille d'un problème peut être exprimée en fonction du nombre de contraintes, de variables, la taille de leurs domaines ou la combinaison de ces trois paramètres. La taille du domaine détermine le nombre de branches à chaque nœud du graphe. Le nombre de variables détermine quant à lui la profondeur de l'arbre de recherche.

II.1.2. La programmation dynamique

La programmation dynamique [RUSTICHINI 98] est une méthode de résolution de problèmes dans laquelle intervient la variable temps, le plus souvent sous forme discrète. Cette approche s'appuie sur un principe d'optimalité[22] selon lequel "toute sous-politique, d'une politique optimale, est optimale". La résolution est basée sur une décomposition du problème en sous-problèmes plus simples. A chaque sous-problème correspond un ensemble d'options, représentant chacune un coût en terme de fonction objectif. Un ensemble de choix doit donc être effectué pour les différents sous-problèmes dans le but d'arriver à une solution optimale.
Les méthodes de la programmation dynamique, tel que l'algorithme A* [PEARL 84], permettent de résoudre un problème en combinant les solutions de ses sous-problèmes. Elles s'adaptent généralement aux problèmes pouvant être découpés en phases (sous-problèmes), i.e., le processus de prise de décision peut être organisé de manière séquentielle.

II.1.3. L'approche de construction

L'approche de construction constitue l'une des approches les plus anciennes, et consiste à construire pas à pas une solution de la forme $s = (<x_1,v_1> <x_2,v_2>...<x_n,v_n>)$, en suivant un ordre statique ou dynamique d'instanciation. Partant d'une solution partielle initialement vide $s =()$, elle cherche à étendre à chaque étape la solution partielle $s = (<x_1,v_1>...<x_{i-1},v_{i-1}>)$ ($i \leq n$) de l'étape précédente en instanciant une variable supplémentaire x_i avec la valeur v_i de son domaine afin de former une nouvelle solution partielle $s' = (<x_1,v_1>...<x_{i-1},v_{i-1}> <x_i,v_i>)$. Ce processus se répète jusqu'à l'obtention d'une solution complète ou que toutes les possibilités soient testées. Durant la recherche d'une solution, une méthode de construction fait intervenir des heuristiques pour effectuer le choix de la variable suivante et de sa valeur. Sa performance dépend donc largement de ces heuristiques utilisées.

Nous présentons dans cette partie quelques méthodes basées sur le principe de construction, en l'occurrence : generate and test, le backtrack chronologique, les stratégies arrière, les stratégies avant et la méthode de branch and bound.

[22] Enoncé par Richard Bellman dans les années cinquante [BELLMAN 57].

II.1.3.1. Generate and test

Il représente l'un des premiers algorithmes conçus pour résoudre le problème de satisfaction de contraintes. L'algorithme "generate and test" consiste à essayer toutes les combinaisons des valeurs admissibles pour les variables en retenant celles qui respectent l'ensemble des contraintes. Dans ce paradigme, chaque combinaison possible de variables est générée et testée pour voir si elle satisfait toutes les contraintes. La vérification du viol d'une contrainte s'effectue donc après l'instanciation complète de toutes les variables du problème. Le nombre de combinaisons considérées par cette méthode est la taille du produit cartésien de tous les domaines des variables.

II.1.3.2. Backtrack chronologique (retour arrière)

Le backtrack ou encore "test and generate", est un algorithme plus performant que "generate and test" et se base essentiellement sur une recherche en profondeur d'abord (depth-first search) réalisant des retours arrières chronologiques [KUMAR 87]. Dans sa version standard, cet algorithme de résolution instancie les variables dans un ordre prédéterminé et procède par essais et erreurs : si une instanciation I_k est consistante, une variable supplémentaire x_{k+1} (la suivante dans l'ordre) est alors instanciée : c'est la phase avant (forward phase) du backtrack. Si aucune valeur du domaine de x_{k+1} ne peut prolonger I_k de manière consistante, un retour est alors effectué sur la dernière variable x_k instanciée, à laquelle sera affectée la valeur suivante dans son domaine : c'est la phase arrière (backward phase). La recherche s'arrête lorsqu'une solution est trouvée (s'il en existe) ou que toutes les possibilités ont été explorées, selon que l'on désire disposer d'une seule solution ou de toutes les solutions. Contrairement à l'algorithme précédent de "test and generate", la vérification du viol d'une contrainte se fait après l'instanciation partielle des variables liées à une contrainte. Quand une instanciation partielle viole une contrainte, le backtrack élimine un sous espace du produit cartésien des domaines de toutes les variables. Nous renvoyons à l'annexe 1 pour un exemple d'algorithme de backtrack.

Outre sa complexité temporelle (exponentielle par rapport à la taille du problème), la méthode de backtrack souffre aussi du "thrashing"[23] à cause de son principe qui consiste à revenir à la dernière variable instanciée qui n'est pas forcément la cause de l'échec. Sa performance peut donc être améliorée par différents moyens :

[23] Thrashing : Ré-exploration de sous-parties de l'espace de recherche déjà rejetées [GASCHNIG 79].

- Intégrer un mécanisme de filtrage[24] au moment de la recherche de solution afin de détecter les échecs et les éliminer de l'espace de recherche. Le problème est ainsi transformé à un problème d'espace de recherche réduit.

- Procéder à des retours arrières intelligents (cas du backtrack intelligent) : Le problème n'est pas transformé à un autre problème mais l'espace de recherche est parcouru de façon plus efficace (plus intelligente). L'information concernant les échecs rencontrés est retenue et exploitée durant la recherche pour éviter de telles situations.

- Choisir un bon ordre d'instanciation de variables, d'affectation de valeurs et de vérification de la satisfaction des contraintes.

La combinaison optimale des différentes techniques dépend du problème traité et constitue le sujet de plusieurs recherches [DECHTER 89].

Certains systèmes de programmation sous contraintes comme par exemple ALICE [LAURIERE 78], CHIP [DINCBAS 88], Prolog III [COLMERAUER 90] ou ILOG Solver [PUGET 94] sont essentiellement fondés sur le principe de retour arrière. Les méthodes qui s'inspirent du backtrack peuvent être classées en 2 catégories [DECHTER 89] : Les stratégies arrière : look-back et les stratégies avant : look-ahead.

II.1.3.3. Les stratégies arrière

Ces stratégies améliorent le processus de recherche en se basant sur l'apprentissage et l'exploitation de l'information trouvée au cours de la recherche. Parmi ces approches, nous trouvons le backtrack intelligent et le backjumping.

a. Le backtrack intelligent

Le backtrack intelligent [STALLMAN 77] est une stratégie générale basée sur l'identification de la vraie cause des échecs (en cas d'inconsistance) afin que l'algorithme puisse revenir sur la variable qui a causé l'échec et de prendre ainsi des décisions pertinentes. Cependant, cette stratégie nécessite un temps de calcul élevé.
Des versions de backtrack intelligent développées pour Prolog pour la résolution des CSPs sont proposées dans [BRUYNOOGHE 84] et [KUMAR 88].

[24] Cette technique sera étudiée dans la partie III.1.4.

b. Le backjumping

A la différence du backtrack chronologique qui retourne en cas d'échec à la dernière variable instanciée, l'algorithme de backjumping [DECHTER 90] saute plusieurs niveaux en arrière jusqu'à la dernière variable liée par une contrainte avec la variable courante (retour à la dernière mauvaise décision). Si cette variable de retour ne possède pas d'autres valeurs, l'algorithme retourne plus loin à la variable la plus récente liée par une contrainte avec la nouvelle variable de retour et ainsi de suite. Une version plus puissante de cet algorithme est appelée conflict-directed backjumping [PROSSER 95] ou graph based backjumping [TSANG 93].

D'autres méthodes existent comme l'assumption-based truth maintenance system [DE KLEER 86a][DE KLEER 86b], le backchecking et le backmarking [TSANG 93].

II.1.3.4. Les stratégies avant ou anticipatives

Ce type de méthodes se base sur la réduction de l'espace de recherche du problème à chaque étape et la détection des insatisfiabilités. Parmi les méthodes développées, le forward checking [HARALICK 80] constitue l'une des méthodes les plus performantes. Contrairement au backtrack chronologique qui vérifie que la valeur de la variable courante est compatible avec les valeurs affectées aux variables précédentes, le forward checking vérifie qu'il existe au moins une valeur pour chaque variable non encore instanciée, qui soit consistante avec la dernière valeur affectée. En d'autres termes, avant d'affecter une valeur à une variable, cette stratégie avant vérifie que cette valeur est compatible avec les variables suivantes. Des labels sont ainsi définis afin de garder la trace des valeurs autorisées pour les variables suivantes. Chaque variable possède un ensemble de valeurs, son label, compatible avec la valeur des variables déjà instanciées. Donc à chaque instanciation, toutes les valeurs inconsistantes sont éliminées des labels des variables suivantes.

D'autres algorithmes existent comme le AC-lookahead ou le directional AC-lookahead [TSANG 93]. Une comparaison entre les performances des algorithmes "generate and test", le backtrack, forward checking, partial lookahead, full lookahead et really lookahead est donnée dans [NADEL 88] qui présente une évaluation détaillée de ces algorithmes dans le contexte des problèmes des N-Reines [KUMAR 92].

II.1.3.5. Séparation et évaluation progressive (Branch and Bound)

Le principe de cette méthode de construction [LAWLER 96][PAPADIMITRIOU 82] repose sur l'utilisation d'heuristiques capables d'estimer les meilleures valeurs (par rapport à la fonction d'évaluation) de toutes les possibilités d'affectation des variables sous la branche courante de l'arbre de recherche. L'utilisation d'heuristiques fiables permet de tailler l'espace de recherche qui ne contient pas la solution optimale et d'éviter ainsi une énumération coûteuse de toutes les configurations possibles.

L'application de cette technique qui convient parfaitement à la résolution des CSOPs, nécessite une fonction heuristique h qui affecte une valeur numérique (appelée h-value) à chaque tuple t de valeurs. A chaque nœud de la recherche, deux valeurs sont calculées : la f-value qui représente la valeur réelle de l'instanciation partielle de la racine jusqu'au nœud courant et la h-value correspondant à la valeur estimée de l'instanciation du nœud suivant jusqu'au dernier nœud du sous-arbre. Dans les problèmes de maximisation/minimisation et pour que la fonction heuristique utilisée soit admissible, il faut que la h-value de chaque tuple soit une sur-estimation/sous-estimation (la meilleure valeur possible) de la valeur réelle (f-value) de chaque tuple de solution qui projette au tuple t. Une variable globale appelée bound, est initialisée à moins l'infini dans les problèmes de maximisation (plus l'infini dans les problèmes de minimisation). A la différence du backtrack, avant d'instancier une nouvelle variable et étendre ainsi une affectation partielle, la h-value est calculée pour la solution partielle courante. Si cette valeur estimée est moins bonne que la variable bound alors le sous-arbre situé sous l'affectation partielle courante dans le graphe de contraintes est taillé de l'espace de recherche puisqu'il est estimé qu'il n'aura pas de chance de contenir une solution meilleure que celle trouvée auparavant. Dans le cas contraire, le processus continue avec l'instanciation d'une nouvelle variable et ainsi de suite. Dès qu'une solution est trouvée, son évaluation réelle f-value remplace l'ancienne valeur de bound si elle est meilleure que celle ci. La qualité d'un algorithme branch and bound est déterminée donc par la qualité de la fonction heuristique et le choix de la borne adéquate (bound) au début de l'algorithme.

Cette méthode garantit la complétude si la fonction heuristique h retourne la borne supérieure (ou inférieure) de f-value. Dans le cas contraire, la fonction heuristique peut sous-estimer la valeur réelle d'un tuple et éliminer ainsi une partie de l'espace de recherche qui peut contenir une solution optimale et la solution fournie sera par conséquent sous-optimale.

Un exemple typique de branch and bound est l'algorithme A*[25] avec une stratégie «meilleur d'abord» pour la recherche d'un plus court chemin dans un graphe valué [PEARL 84].

II.1.4. Les méthodes de pré-traitement

Plusieurs techniques de pré-traitement [KUMAR 92] ont été étudiées afin d'accroître l'efficacité de résolution des problèmes faisant intervenir des contraintes. Ces techniques sont en général utilisées en combinaison avec une autre méthode de recherche de solution. Les techniques de filtrage, en l'occurrence le filtrage par consistance locale, représentent les méthodes de pré-traitement les plus utilisées. L'idée de la consistance est d'éliminer les valeurs qui ne peuvent jamais faire partie d'une solution (éliminer des inconsistances partielles) et peut être appliquée à n'importe quelle étape de la recherche. Si les domaines des variables ou des contraintes sont réduits à l'ensemble vide par l'application des techniques de filtrage, alors le problème est insolvable.

Le filtrage par consistance locale ou encore la propagation de contraintes consiste à supprimer de chaque domaine toutes les valeurs sans support. Une valeur v_i d'un domaine $dom(x_i)$ a un support v_j dans le domaine $dom(x_j)$ si le couple (v_i, v_j) est autorisé par la contrainte liant les variables x_i et x_j.
La consistance des nœuds [MACKWORTH 77] est vérifiée si toutes les valeurs de toute variable x_i respectent toutes les contraintes unaires $C(x_i)$ sur cette variable. Les valeurs qui ne respectent pas une contrainte unaire sur x_i sont éliminées de son domaine. Par application itérative de la consistance des nœuds sur toutes les contraintes, nous obtenons un réseau qui satisfait la consistance d'arc. Etablir la consistance d'arc dans un réseau de contraintes <X,D,C> consiste donc à produire un réseau de contraintes <X,D',C> équivalent au réseau d'origine et vérifiant la propriété de la consistance d'arc. D'une façon générale, toute instanciation consistante d'une variable peut se prolonger en une instanciation consistante de k variables (propriété de k-consistance). Un problème vérifie la k-consistance si toute instanciation consistante de k-1 variables peut se prolonger en une instanciation consistante de k variables. L'obtention de la propriété de k-consistance par suppression des valeurs ou combinaisons de valeurs localement inconsistantes s'appelle filtrage par k-consistance [FREUDER 78].

[25] L'algorithme A* construit l'arborescence en évaluant a priori les chances de trouver la solution optimale dans une branche particulière.

Un problème de n variables vérifiant toutes les propriétés de 2, 3, ..., n-consistance possèderait nécessairement une solution. Malheureusement, le coût d'obtention de ces propriétés, qui croît exponentiellement avec le niveau de consistance voulu, est plus élevé que les méthodes classiques de recherche [KUMAR 92]. Des expériences menées par différents chercheurs sur des problèmes variés [DECHTER 89][GASCHNIG 78][GASCHNIG 79][HARALICK 80][McGREGOR 79] et [PROSSER 91] ont montré qu'il vaut mieux appliquer la propagation de contraintes dans une forme limitée. A cet effet, au lieu d'effectuer un filtrage systématique à chaque nœud de l'arbre de recherche, il est possible d'effectuer un filtrage « opportuniste » tel est le cas des algorithmes de mémorisation de contraintes appelés aussi algorithmes d'apprentissage [DECHTER 86]. Ces algorithmes identifient et mémorisent, pendant la recherche, des tuples localement cohérents mais qui ne peuvent s'étendre à une solution [DECHTER 90].

Plusieurs versions d'algorithmes de consistance locale ont été proposées : AC-1, AC-2, AC-3 [MACKWORTH 77], algorithme de Waltz (équivalent AC-2) [WALTZ 75], AC-4 [MOHR 86], AC-5 [DEVILLE 91] AC-6 [BESSIERE 94], AC-7 [BESSIERE 95], LAC-7 [SCHIEX 96], AC-2000 et AC-2001 [BESSIERE 01].

II.1.5. Orientation de la recherche et amélioration de l'efficacité

La recherche peut être rendue plus efficace par le choix d'un bon ordre d'instanciation des variables [FOX 89], des valeurs des variables [MINTON 92] et des contraintes à examiner. Le choix de l'ordre des contraintes influe sur l'efficacité de l'algorithme de résolution à cause du coût de calcul nécessaire pour la vérification de la satisfaction d'une contrainte. En général, les heuristiques portent plus souvent sur le choix de l'ordre des variables et des valeurs plutôt que sur le choix de l'ordre des contraintes à satisfaire car les informations disponibles concernant les deux premiers points semblent plus riches. Nous présentons dans cette partie l'ordonnancement des variables et des valeurs. L'ordre de satisfaction des contraintes quant à lui fera l'objet d'une étude détaillée dans le chapitre 4.

II.1.5.1. L'ordonnancement des variables

Des expériences menées par différents chercheurs ont montré que l'ordonnancement dans lequel des variables sont choisies pour instanciation peut avoir un impact important sur l'efficacité des algorithmes de recherche [BITNER 75][FREUDER 85]. En effet, un bon ordre

de variables déplace les échecs à des niveaux supérieurs de l'arbre de recherche. L'ordonnancement des variables peut être statique ou dynamique :

- *L'ordonnancement statique* : L'ordre des variables est déterminé avant le début de la recherche. Une heuristique qui convient au backtrack simple consiste à classer les variables suivant leur implication dans des contraintes.

- *L'ordonnancement dynamique* : Le choix de la prochaine variable à instancier à chaque étape dépend de l'état courant de la recherche. Ce type d'ordonnancement n'est pas faisable avec tous les algorithmes, e.g., pour le simple backtrack, il n'y a pas d'information particulière générée pendant la recherche qui peut être utilisée pour faire un ordonnancement différent de celui de départ. Plusieurs heuristiques pour l'ordonnancement dynamique ont été développées, en l'occurrence :

  - *Min-width ordering* qui consiste à choisir la variable, non encore instanciée, la moins liée à des contraintes avec les variables non encore instanciées.

  - *Max-degree ordering* [FREUDER 85] qui consiste à instancier la variable, non encore instanciée, la plus connectée dans le graphe original (la plus liée par des contraintes).

  - *Search-rearrangement method* qui sélectionne la variable avec le minimum d'alternatives possibles restantes. Cette heuristique développée dans [BITNER 75] est souvent utilisée avec forward checking.

II.1.5.2. L'ordonnancement des valeurs (value ordering)

Une fois que la décision est prise concernant la variable à instancier, il peut y avoir plusieurs valeurs possibles. L'ordre dans lequel ces valeurs sont considérées peut avoir un grand effet sur le temps nécessaire pour trouver la première solution. Un bon ordre d'affectation des valeurs déplace la solution du problème à gauche de l'arbre afin qu'elle soit trouvée rapidement. Dans ce cadre, si le problème a une solution, et si une valeur correcte est affectée à chaque variable, alors une solution peut être trouvée sans avoir besoin d'un algorithme de recherche. Cependant, si toutes les solutions sont exigées ou s'il n'y a aucune solution, le classement des valeurs est indifférent. L'ordonnancement des valeurs est souvent utilisé en conjonction avec une stratégie de recherche avant. Parmi les heuristiques utilisées, nous citons :

- *Succeed first* : Son principe consiste à choisir la valeur qui a plus de chance de réussir (la moins probable d'amener à un échec).

- *Min-Conflict* : Cette heuristique peut être utilisée conjointement pour l'ordonnancement des variables et des valeurs. Elle consiste à choisir aléatoirement à chaque itération, une variable en conflit, c'est-à-dire une variable impliquée dans une contrainte non satisfaite [MINTON 90]. Pour cette variable, la valeur qui minimise le nombre de conflits (fonction de coût) est choisie, avec choix aléatoire en cas d'égalité. L'heuristique min-conflict, utilisée pour l'ordonnancement et la planification du télescope spatial Hubble[26], a prouvé son efficacité sur des problèmes d'ordonnancement [MINTON 92].

II.1.6. Synthèse des méthodes exactes

Les techniques exactes, mieux adaptées à la résolution des problèmes d'affectation sous contraintes n'accordent pas suffisamment d'intérêt à la modélisation des contraintes et leur première préoccupation consiste à choisir ou développer la bonne technique permettant de satisfaire ces contraintes. En plus, les contraintes sont généralement supposées être clairement définies initialement, ce qui n'est pas souvent le cas. Le décideur exprime ses besoins, souvent de façon imprécise ou incertaine et nécessite un minimum d'assistance de la part de l'homme d'étude pour l'aider à mieux exprimer et clarifier ses besoins et ses préférences. Ses contraintes sont susceptibles d'évoluer au cours du processus de résolution et son implication dans ce processus contribue à sa convergence psychologique/mentale (mieux clarifier et définir ses contraintes) [COURBON 93].

Les méthodes exactes sont principalement utilisées quand l'optimalité est primordiale ou lorsque toutes les solutions du problème doivent être trouvées. Cependant, les problèmes d'optimisation peuvent atteindre rapidement une grande complexité et nécessiter des temps de calcul considérables à cause de l'explosion combinatoire du nombre des solutions à envisager. Les méthodes approchées constituent une alternative très intéressante pour traiter ce genre de problèmes où les méthodes exactes se révèlent peu efficaces.

II.2. Les méthodes approchées

Les méthodes approchées ou incomplètes, utilisées essentiellement pour les problèmes de grande taille ou comportant peu de contraintes, sont basées sur l'exploration itérative de l'espace de recherche pour la découverte d'une solution de bonne qualité en un temps de calcul raisonnable. Un des avantages des méthodes approchées réside donc dans la possibilité

[26] Site Web : http://hubble.stsci.edu/.

de contrôler le temps de calcul. En effet, contrairement aux méthodes exactes qui ne peuvent fournir une solution complète au problème avant l'instanciation de toutes les variables, le processus de recherche dans les méthodes approchées[27] peut être arrêté à tout moment tout en obtenant une solution complète au problème. Toutefois, du fait de leur aspect itératif, la qualité des résultats fournis par les heuristiques dépend généralement du temps d'exécution et la probabilité d'avoir une solution meilleure augmente au cours du temps (sans toutefois garantir une solution meilleure).

Les méthodes approchées sont plus adaptées aux problèmes peu contraints. Leur application sur des problèmes très contraints nécessite un effort particulier de codage et d'adaptation selon deux façons :

- Ajuster les opérateurs et les mécanismes propres à chaque méthode pour mieux contrôler la satisfaction des contraintes et générer ainsi des solutions satisfaisantes à chaque étape de la résolution.

- Intégrer une fonction de pénalité dans la fonction d'évaluation afin de pénaliser, en fonction du nombre et de l'importance des contraintes violées, la valeur d'évaluation relative à chaque solution trouvée.

Depuis une dizaine d'années, des progrès importants ont été réalisés avec l'apparition d'une nouvelle génération de méthodes approchées puissantes et générales, appelées méta-heuristiques[28]. Contrairement aux algorithmes traditionnels dédiés à un problème spécifique comme la programmation linéaire ou la programmation dynamique, les méta-heuristiques constituent des mécanismes très généraux qui peuvent être adaptés pour traiter de nombreux problèmes différents et concevoir des méthodes heuristiques adaptées à chacun de ces problèmes [CHARON 95]. Une heuristique est une règle ou logique d'action, applicable à certaines situations, qui permet la plupart du temps d'aboutir plus rapidement à la solution sans garantir l'optimalité ni l'admissibilité. Une méta-heuristique est donc définie de manière similaire, mais à un niveau d'abstraction plus élevé.

La plupart des méta-heuristiques font appel à une structure de voisinage et/ou à une sorte de mémorisation (mémoire). Cependant, il n'existe en général pas de résultats théoriques

[27] A l'exception de certaines approches de construction supposées approchées, en l'occurrence l'algorithme glouton.
[28] Le terme « méta-heuristique » a été initialement utilisé par F. Glover [GLOVER 86] pour distinguer la méthode tabou des heuristiques spécifiques.

garantissant la convergence d'une méthode approchée vers un optimum global. La raison principale de cet état de fait tient à la nature même de ces méthodes. En effet, à l'exception de quelques recherches déterministes avec la méthode tabou, les méta-heuristiques sont souvent basées sur des choix probabilistes et sont donc non déterministes[29] car elles utilisent un générateur pseudo-aléatoire. Nous présentons dans cette partie une liste non exhaustive des méta-heuristiques les plus utilisées.

II.2.1. Les méthodes de construction

Certaines méthodes de construction ne garantissent pas la complétude et sont donc considérées comme approchées. Parmi ces méthodes, l'algorithme glouton constitue l'une des techniques les plus rapides. Les algorithmes qui résolvent les problèmes d'optimisation parcourent en général une série d'étapes, au cours desquelles ils sont confrontés à un ensemble d'options. Un algorithme glouton fait toujours le choix optimal localement, dans l'espoir que ce choix mènera à la solution optimale globalement. Contrairement à la programmation dynamique où le choix dépend de la solution de sous-problèmes, dans un algorithme glouton, le choix qui semble le meilleur sur le moment est effectué (sans remettre en cause les choix effectués précédemment) avant de s'attaquer à la résolution des sous-problèmes qui en résultent.
Les méthodes gloutonnes sont généralement rapides, mais fournissent le plus souvent des solutions de qualité médiocre et ne garantissent l'optimum que dans des cas particuliers [VINCE 02].

II.2.2. Les méthodes de voisinage

Un voisinage est l'ensemble des solutions obtenues à partir d'une solution donnée en effectuant quelques transformations simples. Il peut être représenté à l'aide d'un graphe orienté où les nœuds représentent les solutions et les arcs représentent les liens de voisinage.

Définition [HAO 99] : Soit X l'ensemble des configurations admissibles d'un problème, on appelle voisinage toute application $N : X \rightarrow 2^x$. On appelle mécanisme d'exploration du voisinage toute procédure qui précise comment la recherche passe d'une configuration $s \in X$ à une configuration $s' \in N(s)$. Une configuration s est un optimum local par rapport au voisinage N si f(s) est meilleure que f(s') pour toute configuration $s' \in N(s)$.

[29] Elles donnent un résultat différent à chaque exécution.

Une méthode de voisinage [AHUJA 02], appelée aussi méthode de réparation ou de recherche locale, est un processus itératif fondé sur deux éléments essentiels : un voisinage et une procédure exploitant le voisinage. Contrairement à l'approche de construction traitant des instanciations partielles, i.e., à chaque itération, seulement un sous-ensemble de variables est instancié, les méthodes de voisinage manipulent des configurations complètes où toutes les variables sont instanciées. Une méthode typique de voisinage débute avec une configuration initiale, et réalise ensuite un processus itératif qui consiste à remplacer la configuration courante par l'une de ses voisines en tenant compte d'une fonction de coût. Ce processus s'arrête et retourne la meilleure configuration trouvée quand la condition d'arrêt est réalisée. Cette condition d'arrêt concerne généralement une limite pour le nombre d'itérations (ou le temps de calcul) ou un objectif à réaliser.

Les méthodes de voisinage diffèrent essentiellement entre elles par les processus de parcours et de sélection de voisinage. Parmi les méthodes les plus utilisées, nous trouvons l'amélioration itérative, le recuit simulé, les algorithmes d'acceptation avec seuil et la recherche tabou.

II.2.2.1. L'amélioration itérative

L'amélioration itérative [AARTS 97], appelée aussi méthode de descente ou de recherche locale[30], consiste à choisir à chaque itération un voisin qui améliore la configuration courante. Deux choix sont possibles : choisir le premier voisin qui améliore strictement la fonction d'optimisation ou bien énumérer tous les voisins afin d'en découvrir le meilleur. Cette dernière solution est certainement plus coûteuse, mais le voisin découvert sera en général de meilleure qualité. Comme l'espace des solutions X est fini, cette procédure de descente s'arrête toujours, et la dernière configuration trouvée ne possède pas de voisin strictement meilleur qu'elle-même. L'avantage principal de cette méthode réside dans sa grande simplicité et sa rapidité mais les solutions produites sont souvent de qualité médiocre et de coût très supérieur au coût optimal. Pour remédier à ce problème, une solution simple, appelée méthode de relance aléatoire, consiste à générer une nouvelle configuration de départ de façon aléatoire et à recommencer une nouvelle descente. Cependant, cette solution n'exploite pas les optima locaux déjà découverts. Une autre solution consiste à accepter des voisins de même performance que la configuration courante, ce qui permet à la recherche de se déplacer sur les

[30] Dans la littérature, le terme «recherche locale» est de plus en plus employé pour désigner toute la classe des méthodes de voisinage et pas seulement l'amélioration itérative.

plateaux, mais cette méthode n'est pas suffisante pour ressortir de tous les optima locaux. D'autres raffinements plus élaborés sont également possibles, par exemple, l'introduction de voisinages variables [KERNIGHAN 70] et les techniques de réduction ou d'élargissement [LIN 73].

II.2.2.2. Le recuit simulé (Simulated Annealing)

Le recuit simulé [KIRKPATRICK 83] constitue, parmi les méthodes de voisinage, l'une des plus anciennes et des plus populaires. Il a acquis son succès essentiellement grâce à des résultats pratiques obtenus sur de nombreux problèmes NP-difficiles, tel est le cas des applications présentées dans [BONOMI 84][COLLINS 88][KOULAMAS 94].
La méthode du recuit simulé s'inspire du processus de recuit physique utilisé en métallurgie pour améliorer la qualité d'un solide (la structure d'un solide refroidi dépend de la vitesse de refroidissement). Partant d'un état initial à haute température (à laquelle le solide est devenu liquide), la phase de refroidissement conduit la matière liquide à retrouver sa forme solide par une diminution progressive de la température. La température est contrôlée par une fonction décroissante qui définit un schéma de refroidissement. Chaque température est maintenue jusqu'à ce que la matière trouve un équilibre thermodynamique. Le processus du recuit simulé répète une procédure itérative, qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût. En pratique, l'algorithme s'arrête et retourne la meilleure configuration trouvée lorsque aucune configuration voisine n'a été acceptée pendant un certain nombre d'itérations à une même température ou lorsque la température atteint la valeur zéro.

Toute nouvelle configuration est obtenue en faisant subir une modification aléatoire à un composant quelconque. Soit ΔE la différence d'énergie ou de coût occasionnée par une telle perturbation, la nouvelle configuration est acceptée systématiquement si l'énergie du système diminue ($\Delta E \leq 0$). Sinon, elle est acceptée avec une certaine probabilité $p(\Delta E,T)$[31] où T est la température courante. L'acceptation ou non d'une nouvelle configuration dont l'énergie est supérieure à celle de la configuration courante est déterminée de manière probabiliste : un réel $0 \leq \theta < 1$ est tiré aléatoirement et ensuite comparé avec $p(\Delta E,T)$. Si $\theta \leq p(\Delta E,T)$, alors la

[31] Une façon de calculer une probabilité en fonction d'un facteur température et d'une distance entre deux états (configurations) est la suivante : $p(\Delta E,T) = e^{-\Delta E/(K*T)}$ où k est une constante physique connue sous le nom de constante de Boltzmann $k=1{,}380662\ 10^{-23}$ joules/degré (mais sa valeur peut être ajustée selon l'application).

nouvelle configuration est acceptée pour remplacer la configuration courante, sinon, la configuration courante est maintenue et réutilisée pour générer une nouvelle configuration.
Après un grand nombre de perturbations, un tel processus fait évoluer le système vers un état d'équilibre thermodynamique selon la distribution de Boltzmann qui est définie par la probabilité de se trouver dans un état d'énergie E : $p(E) = c(T) \times e^{-E/(K*T)}$ où c(T) est un facteur de normalisation.

La performance du recuit simulé dépend largement du schéma de refroidissement utilisé. De nombreux schémas théoriques et pratiques de refroidissement [HAJEK 88] ont été proposés et peuvent être classés en trois catégories :

- Réduction par paliers : Chaque température est maintenue pendant un certain nombre d'itérations et décroît ainsi par paliers.

- Réduction continue [LUNDY 86] : La température est modifiée à chaque itération.

- Réduction non-monotone [CONNOLLY 90] : La température décroît à chaque itération avec des augmentations occasionnelles.

Nous proposons en annexe 1, un exemple d'algorithme de recuit simulé adapté à notre problématique.

II.2.2.3. Les algorithmes d'acceptation avec seuil (Threshold algorithms)

Ces algorithmes sont des variantes du recuit simulé et la différence se situe essentiellement au niveau de l'acceptation de dégradation à chaque étape. Pour le recuit simulé, cette décision est prise selon un critère de probabilité qui dépend de la température et d'un nombre aléatoire. Dans un algorithme d'acceptation avec seuil [DUCEK 90], une telle décision est prise de manière déterministe. A chaque itération k, l'acceptation d'un voisin s' se base uniquement sur une fonction auxiliaire f(s',s) et un seuil T_k : s' est accepté si $f(s',s) < T_k$. La fonction f(s',s) et le seuil T_k peuvent être définis de nombreuses manières. Dans le cas le plus simple, la fonction f(s',s) est définie par la différence de coût. Le paramètre de seuil est défini de manière analogue à la température T du recuit simulé : Il est initialisé à une valeur élevée puis décroît progressivement à chaque fois qu'un nombre prédéterminé d'itérations est effectué, dans le but de diminuer progressivement la chance d'accepter des configurations qui dégradent la fonction de coût. Ces algorithmes ont été utilisés avec succès dans de nombreux problèmes [DUCEK 93].

II.2.2.4. La recherche tabou

A l'inverse du recuit simulé qui génère de manière aléatoire une seule solution s' voisine de s à chaque itération, la méthode tabou [GLOVER 86] examine un échantillonnage de solutions et retient la meilleure s', même si s' est plus mauvaise que s. La recherche tabou ne s'arrête donc pas au premier optimum trouvé. Cependant, cette stratégie peut entraîner des cycles (par exemple un cycle de longueur 2 : s $\rightarrow$s'$\rightarrow$s $\rightarrow$s'). Pour empêcher de telles situations, les k dernières configurations[32] explorées sont retenues dans une mémoire à court terme, appelée la liste tabou, afin d'interdire les mouvements qui conduisent à l'une de ces configurations.

Afin d'éviter une mémorisation trop coûteuse de configurations entières, lorsqu'un mouvement vient d'être effectué, la liste tabou mémorise seulement la caractéristique perdue par la configuration courante. Cependant, non seulement la configuration courante ne pourra pas réapparaître lors des k prochaines itérations, mais de nombreuses autres configurations peuvent être également interdites. En d'autres termes, lorsque les listes "tabou" font intervenir des caractéristiques de modifications, les interdictions qu'elles engendrent peuvent s'avérer trop fortes et restreindre l'ensemble des solutions admises à chaque itération d'une manière jugée trop brutale. Un mécanisme particulier, appelé l'aspiration [GLOVER 97] permet de lever le statut tabou d'une configuration, sans pour autant introduire un risque de cycles dans le processus de recherche. La fonction d'aspiration peut être définie de plusieurs manières. La façon la plus simple consiste à révoquer le statut tabou d'un mouvement si ce dernier permet d'atteindre une solution de qualité supérieure à celle de la meilleure solution trouvée jusqu'à lors.

Il existe également d'autres techniques intéressantes pour améliorer la puissance de la méthode tabou, en particulier, l'intensification et la diversification. Toutes les deux se basent sur l'utilisation d'une mémoire à long terme et se différencient selon la façon d'exploiter les informations de cette mémoire.

L'intensification se base sur l'idée d'apprentissage de propriétés favorables : les propriétés communes souvent rencontrées dans les meilleures configurations visitées sont mémorisées au cours de la recherche, puis favorisées pendant la période d'intensification. Une autre manière d'appliquer l'intensification consiste à mémoriser des points de retour correspondant à des solutions de bonne qualité et à retourner vers une des ces solutions.

[32] La valeur de k dépend du problème à résoudre et peut éventuellement évoluer au cours de la recherche.

La diversification a un objectif inverse de l'intensification : elle cherche à diriger la recherche vers des zones inexplorées. Sa mise en oeuvre consiste souvent à modifier temporairement la fonction de coût pour favoriser des mouvements n'ayant pas été effectués ou à pénaliser les mouvements ayant été souvent répétés.

La méthode tabou suscite un intérêt toujours croissant depuis sa découverte et de nombreux raffinements ont été introduits dans la méthode [GLOVER 95]. Des résultats pratiques intéressants ont été obtenus pour de nombreux problèmes d'optimisation combinatoire et d'affectation sous contraintes [GLOVER 97]. Cette technique fait partie des meilleures méta-heuristiques pour ces problèmes et se caractérise par sa stratégie agressive de recherche (choix d'un des meilleurs mouvements à chaque itération) et une possibilité d'adaptation et d'intégration des connaissances spécifiques des problèmes étudiés.

II.2.3. Les algorithmes évolutifs

Les algorithmes évolutifs s'inspirent du processus d'évolution naturelle en analogie avec les mécanismes d'évolution des espèces vivantes. L'évolution peut être vue comme une méthode produisant de nouvelles solutions en fonction des changements de l'environnement. Un algorithme évolutif typique est composé de trois éléments essentiels :

- *Une population* constituée de plusieurs individus, souvent générés aléatoirement, représentant chacun une solution potentielle (configuration) du problème. Une population initiale diversifiée est gage d'une solution de qualité, sinon le processus de résolution risque de se trouver avec des espèces dégénérées et ne jamais s'approcher de la solution optimale.

- *Un mécanisme d'évaluation de l'adaptation* (fitness) de chaque individu de la population à l'égard de son environnement extérieur.

- *Un mécanisme d'évolution* composé d'opérateurs permettant de sélectionner certains individus pour se reproduire donnant ainsi naissance à d'autres générations d'individus. Selon l'analogie de l'évolution naturelle, la qualité des individus de la population devrait tendre à s'améliorer au fur et à mesure du processus. En effet, un algorithme évolutif comporte trois opérateurs essentiels, à savoir la sélection, le croisement et la mutation :

  - *La sélection* a pour objectif de choisir les individus qui vont pouvoir survivre et/ou se reproduire pour transmettre leurs caractéristiques à la génération suivante. Cet opérateur se base généralement sur le principe de conservation des individus les mieux adaptés et d'élimination des moins adaptés. Cependant, il ne faut pas complètement écarter les

individus moins bons, car ils peuvent posséder un élément génétique intéressant et aussi afin d'éviter la convergence prématurée (les meilleurs individus colonisent la population très rapidement).

- *Le croisement* ou recombinaison (crossover) cherche à combiner les caractéristiques des individus parents pour créer des individus enfants avec de nouvelles potentialités dans la génération future. Il sert ainsi d'opérateur d'exploitation des caractéristiques des générations existantes en essayant d'améliorer encore les meilleurs individus déjà produits mais trop d'exploitation peut limiter localement la recherche et converger ainsi vers un optimum local. Le taux de crossover est en général assez fort et se situe entre 70% et 95% de la population totale.

- *La mutation* quant à elle, consiste à changer aléatoirement (avec une probabilité différente) la valeur de certaines variables dans un individu. Elle sert d'opérateur d'exploration pour échapper aux optimums locaux et ce en effectuant de légères modifications de certains individus afin d'obtenir des échantillonnages de régions inconnues. Dans les stratégies basées sur trop d'exploration, le système risque de ne pas converger (du moins pas rapidement). A la différence du crossover, le taux de mutation est généralement faible et se situe entre 0.5% et 1% de la population totale afin d'éviter une dispersion aléatoire de la population.

Bien que les opérateurs soient sensiblement différents, l'individu et la fonction d'adaptation dans un algorithme évolutif, correspondent respectivement à la notion de configuration et à la fonction d'évaluation dans les méthodes de voisinage. La notion de mécanisme d'évolution est proche de celle du mécanisme de parcours du voisinage : Chaque itération d'une méthode de voisinage peut être interprétée comme un mécanisme d'évolution portant sur une population réduite à un seul individu.

La famille des algorithmes évolutifs comporte quatre classes principales, à savoir les algorithmes génétiques, la programmation évolutive, la programmation génétique et les stratégies d'évolution. Ces techniques très proches se différencient essentiellement par leur manière de coder les données du problème ainsi que leur façon de faire évoluer la population d'une génération à l'autre.

II.2.3.1. Les algorithmes génétiques

Il n'existe pas de version unique des algorithmes génétiques et plusieurs versions ont été développées dans la littérature [GOLDBERG 89][HOLLAND 92]. Les algorithmes génétiques classiques introduits par Holland [HOLLAND 75] s'appuient fortement sur un codage universel des individus de la population sous forme de chaînes 0/1 de longueur fixe et un ensemble d'opérateurs génétiques. La mutation est considérée comme un opérateur secondaire par rapport au croisement et doit permettre d'atteindre tout l'espace de recherche.
Les opérateurs génétiques sont définis de manière à opérer aléatoirement sur un ou deux individus sans aucune connaissance sur le problème. Cependant, pour être efficace en optimisation, il est indispensable d'intégrer des connaissances du problème [DAVIS 91] [GREFENSTETTE 87] afin de mieux guider la recherche et la rendre plus efficace. A cet effet, il est possible de combiner le cadre génétique avec d'autres méthodes de résolution pour former les méthodes hybrides ou de spécialiser l'algorithme génétique classique en adaptant (au lieu d'utiliser des opérateurs aléatoires) la mutation et le croisement avec les connaissances spécifiques du problème traité. Un algorithme inspiré du principe d'algorithmes génétiques est donné en annexe 1.

En pratique, les algorithmes génétiques nécessitent un effort considérable pour la codification du problème et sont souvent coûteux en temps de calculs. La mise en oeuvre de ces algorithmes sur un cas réel reste délicate car de nombreux paramètres doivent être fixés tels que la génération de la population initiale, la taille de la population, les probabilités de croisement et de mutation, etc. En effet, en tant que méthode globale d'optimisation, un algorithme génétique classique se base uniquement sur des opérateurs «aveugles» et il est donc rarement en mesure de produire des résultats comparables à ceux d'une méthode de voisinage [HAO 99]. Par contre, les algorithmes génétiques spécialisés ou hybrides ont été utilisés avec succès pour de nombreuses applications dans des domaines très variés.

II.2.3.2. La programmation évolutive

La programmation évolutive [FOGEL 92] suppose que la caractéristique principale de l'intelligence est la capacité d'adaptation comportementale d'un organisme à son environnement. Cette approche s'appuie sur un codage approprié du problème à résoudre et sur les opérations de mutation adaptées au codage. Un cycle d'évolution typique pour la programmation évolutive consiste à copier chaque configuration de la population courante

dans une nouvelle population. Les configurations sont ensuite mutées, conduisant à des nouvelles configurations qui entrent dans une étape de compétition pour survivre dans la génération suivante. La mutation est donc considérée comme un opérateur principal qui doit produire des configurations valides.

II.2.3.3. La programmation génétique

La programmation génétique est basée sur le même principe d'évolution que celui des algorithmes génétiques mais les opérateurs de croisement et de mutation sont un peu différents et travaillent directement sur des structures d'arbres syntaxiques (du problème). L'opérateur de croisement consiste à échanger des sous-arbres issus de nœuds tirés au hasard. La mutation quant à elle consiste à choisir au hasard un nœud de l'arbre et à remplacer le sous-arbre issu de ce nœud par un arbre généré au hasard [D'HAESELEER 94].

II.2.3.4. Les stratégies d'évolution

Les stratégies d'évolution ou stratégies évolutives [BÄCK 96] sont conçues pour résoudre des problèmes d'optimisation continus. Elles se basent essentiellement sur les opérateurs de sélection et de mutation et les individus représentent des points (vecteurs de réels).

L'algorithme le plus simple, noté (1+1)-ES, manipule un seul individu. A chaque itération, l'algorithme génère par mutation un individu enfant à partir de l'individu parent et sélectionne l'un ou l'autre pour le conserver dans la population (selon l'adaptation de chaque individu). La condition d'arrêt est souvent définie par le nombre d'itérations, le temps de calcul réalisé ou l'écart entre deux individus de deux itérations successives. La mutation dans un tel algorithme est aléatoirement appliquée à tous les composants de l'individu pour produire un enfant.

Cet algorithme (1+1)-ES se généralise en un algorithme (m+l)-ES qui signifie que m parents génèrent l enfant à chaque génération (itération) et qu'une sélection ramène ensuite la population de m+l individus à m individus [BÄCK 91]. La recombinaison a été également introduite dans ces algorithmes [BÄCK 93].

II.2.4. Les méthodes hybrides

Plusieurs études ont montré que la mise en commun des principes de plusieurs méthodes (exactes ou approchées) peut mener à l'élaboration d'heuristiques encore plus performantes appelées méthodes hybrides. Le mode d'hybridation qui semble le plus fécond concerne la

combinaison entre les méthodes de voisinage et d'évolution [GLOVER 95]. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche de méthodes de voisinage et d'évolution des algorithmes évolutifs[33].

Parmi les méthodes hybrides, nous trouvons la méthode scatter search [LAGUNA 02], la méthode de colonie de fourmis [DORIGO 99] et GRASP [FEO 95].

II.2.4.1. La méthode scatter search

La méthode scatter search [GLOVER 00][LAGUNA 02] consiste à construire un ensemble de solutions de bonne qualité, à sélectionner des sous-ensembles de ces solutions et à appliquer un opérateur de recombinaison à chacun de ces sous-ensembles. Nous retrouvons ainsi certains principes des algorithmes génétiques.

II.2.4.2. La méthode de colonie de fourmis

La méthode de colonie de fourmis [DORIGO 99] s'inspire du comportement des colonies de fourmis réelles. En effet, malgré la vision très limitée de chaque fourmi, une colonie de fourmis parvient à minimiser la longueur du chemin conduisant à une source de nourriture, grâce aux traces chimiques (phéromones) laissées par chacune des fourmis. Cette méthode se caractérise par la combinaison d'une approche de construction et des mécanismes d'apprentissage fondés sur la mémorisation. Elle consiste à réitérer un algorithme de construction, assimilé à l'action d'une fourmi, dans lequel chacun des choix est déterminé en tenant compte à la fois d'un critère glouton, d'un aspect aléatoire et des traces laissées par les fourmis précédentes.

II.2.4.3. La méthode GRASP

La méthode GRASP (Greedy Randomized Adaptive Search Procedure) introduite dans [FEO 89], est une procédure itérative qui cherche à combiner les avantages des heuristiques gloutonnes, de la recherche aléatoire et des méthodes de voisinage. Un algorithme GRASP répète un processus composé de deux étapes : la construction d'une solution suivie par une descente pour améliorer la solution construite. Durant l'étape de construction, une solution est itérativement construite : chaque itération ajoute un élément dans la solution partielle courante en se basant sur un choix aléatoire de l'élément dans une liste des meilleurs candidats obtenus

[33] Dans le même ordre d'idées, nous pensons que ce type d'hybridation est intéressant pour notre cas applicatif (voir chapitre 5).

avec une fonction gloutonne. Cette étape de construction continue jusqu'à ce qu'une solution complète soit obtenue et sera suivie par une descente pour améliorer la solution construite. Ce cycle comportant les deux étapes de construction et de descente se répète jusqu'à l'obtention d'une solution de bonne qualité. La méthode GRASP a été appliquée avec succès sur plusieurs problèmes d'optimisation [FEO 94][FEO 95].

II.2.5. Synthèse des méthodes approchées

Les techniques approchées se révèlent inefficaces en présence de contraintes multiples et fortes ou face à des problèmes de décision qui ne vérifient pas les conditions d'optimisation. Leur espace d'application se limite souvent aux problèmes de grande taille ou comportant un seul critère à améliorer.
Les applications d'aménagement de territoire ne se limitent généralement pas à un simple critère et intègrent plusieurs critères complexes de nature différente et comportant souvent des aspects non commensurables (e.g., le rôle économique d'une parcelle et son attachement moral pour l'exploitant). Dans de telles situations, l'objectif consiste plutôt à trouver un bon compromis entre les différents critères.

A cet effet, des méthodes récentes et en plein développement, appelées méthodes d'analyse multicritère [TECLE 91][SCHÄRLIG 85] ont vu le jour afin de permettre l'intégration des aspects multiples et le traitement des problèmes multicritères.

II.3. Les méthodes multicritères

Un grand nombre de méthodes multicritères ont été développées afin de structurer la démarche de décision et d'affronter les problèmes multicritères face auxquels les méthodes classiques se trouvent impuissantes. Ces méthodes consistent généralement à rechercher un bon compromis entre les différents critères.
Cependant, certaines faiblesses résultantes du manque de fondements axiomatiques pour la plupart de ces méthodes, rendent difficile le choix d'une méthode à appliquer face à une situation donnée [BOUYSSOU 93]. Ce choix dépend en grande partie des informations contenues dans les critères et des relations de commensurabilité, de dépendance ou de transitivité qui peuvent exister entre elles. Le choix de la méthode multicritère est donc, lui-même, un problème multicritère [LAARIBI 95] et reste dépendant de la nature du problème et des caractéristiques cognitives des acteurs intervenant dans le processus de décision.

Les méthodes multicritères peuvent être classées suivant leurs manières d'évaluer les différentes solutions. En effet, certaines méthodes ne permettent aucune compensation entre les critères, i.e., elles n'acceptent pas des critères dégradés. D'autres, par contre, autorisent la compensation des mauvaises valeurs de certains critères par les bonnes valeurs des autres.

II.3.1. Les méthodes hiérarchiques sans compensation

Ces méthodes [BOUYSSOU 86] consistent à fixer un seuil d'acceptation pour chaque critère et à éliminer à chaque étape, en prenant en compte les critères un par un, les solutions qui ne satisfont pas le critère considéré. Des contraintes supplémentaires, en raison d'une contrainte par critère, sont ainsi ajoutées au système de contraintes initial. Chaque contrainte a pour rôle de cerner l'évolution d'un critère donné par la fixation d'un seuil d'acceptation.
Cependant, la simplicité et la rapidité de cette approche ne doivent pas cacher certaines faiblesses :

- Le processus d'élimination des solutions basé sur les seuils d'acceptation pour chaque critère (conservé / rejeté) risque d'éliminer des bons compromis.

- La difficulté de fixer les seuils de satisfaction. Un seuil très restrictif pour un critère donné, peut déterminer à lui seul le résultat final, notamment lorsqu'une faible proportion de solutions respecte le seuil fixé pour ce critère.

- Mauvaise adaptation aux problèmes sur-contraints : Cette approche qui paraît efficace, restreint fortement l'espace des solutions, notamment quand le problème est déjà sur-contraint et s'avère mieux adaptée aux problèmes sous-contraints.

II.3.2. Les méthodes avec compensation

Ces méthodes supposent d'une manière implicite que la structure de préférence du décideur sur les solutions est compensatoire [BOUYSSOU 86], c'est-à-dire qu'il existe des taux de substitution entre critères. Elles accordent plus de souplesse, notamment dans la résolution des problèmes sur-contraints. Cependant, certains critères peuvent s'améliorer en dépit d'autres et les solutions trouvées risquent d'avoir des critères trop dégradés. Ces solutions peuvent s'avérer impraticables dans la réalité et auront peu de chance d'être acceptées par les décideurs.

Nous distinguons trois approches opérationnelles [ROY 96] : l'agrégation complète transitive, l'agrégation partielle et l'agrégation locale itérative.

II.3.2.1. Approche d'agrégation complète transitive

Dans cette approche d'inspiration américaine[34], les différents critères sont synthétisés dans une seule fonction mathématique monotone (à sens d'évolution unique). A partir des évaluations des différents critères, la fonction d'optimisation résultante, dite d'utilité ou d'agrégation, produit donc une valeur unique évaluant globalement la solution.
Dans certains cas pratiques, cette orientation simple peut s'avérer intéressante ou s'imposer comme la seule alternative envisageable [SCHÄRLIG 85]. Les méthodes relevant de cette approche conviennent bien aux problèmes sous-contraints ou lorsqu'il y a plusieurs critères qui varient de manière continue dans l'espace [PEREIRA 93] ce qui explique leur grande utilisation dans les applications sur les systèmes d'informations géographiques (SIG) pour l'aide à la décision [EASTMAN 99].

Cependant, les méthodes relevant de cette approche révèlent certaines insuffisances :

- La fonction F résultante peut évoluer dans tous les sens, i.e., certains critères peuvent s'améliorer en dépit d'autres. Une solution pour pallier ce problème consiste à implémenter des techniques (heuristiques, seuils) capables de cerner l'espace d'évolution de F et de s'assurer de l'amélioration de tous les critères.

- La difficulté de déterminer la fonction d'agrégation et les poids relatifs aux différents critères.

- Rassembler tous les critères dans une seule fonction d'optimisation revient à une agrégation finale mono-critère. Ceci sous-entend que tous les critères peuvent avoir le même sens d'évolution dans l'espace de recherche, ce qui n'est pas souvent le cas, notamment quand ces critères sont contradictoires, indépendants ou non convertibles à la même unité de mesure. En effet, le passage forcé du multicritère au mono-critère risque d'être incohérent ou d'ignorer certaines informations relatives aux différents aspects du problème. Certains objectifs renferment un aspect non mesurable (aspect moral par exemple) et ne peuvent être exprimés par des critères[35]. L'approche par agrégation partielle (ci-dessous présentée), basée sur la comparaison des solutions deux à deux suivant le même critère, semble pallier ce problème.

[34] Appelée aussi approche du critère unique de synthèse évacuant toute incomparabilité [MAYSTRE 94][ROY 96] ou théorie de l'utilité multi-attribut (MAUT) [VINCKE 89].
[35] Ce point sera étudié en détails dans le chapitre 3.

B. Roy [ROY 93] conclut la présentation de ce type de méthode par la remarque suivante : " La construction d'un critère unique de synthèse apparaît, aux yeux de beaucoup, comme la plus naturelle. On la croit généralement aisée parce que simple. La complexité des résultats théoriques (...), jointe à la pauvreté des résultats concernant divers problèmes importants auxquels se trouve confronté le praticien, nous invite à souligner le caractère trompeur de cette apparente facilité".

Parmi les méthodes d'agrégation complète, nous trouvons la somme, la moyenne pondérée, MAUT (Multiple Attribute Utility Theory) [KEENEY 76], UTA (Utilités additives) [JACQUET 82], SMART, TOPSIS, AHP, etc.

II.3.2.2. Approche d'agrégation partielle

Cette approche[36] se base sur la comparaison des solutions deux à deux afin de construire, entre elles, une ou plusieurs relations binaires, appelées de sur-classement ou de dominance. Cette comparaison s'effectue en considérant, critère par critère, les avantages et inconvénients d'une solution vis-à-vis de l'autre et conduit à déterminer si selon certaines conditions préétablies l'une des deux solutions surclasse (est au moins aussi bonne que) l'autre. Une solution S_1 surclasse (ou domine) S_2 si, par exemple, S_1 est meilleure que S_2 sur au moins un critère tout en étant au moins aussi bonne qu'elle sur tous les autres critères.
Cette étape aboutit à un graphe des relations entre les solutions, qui permet suivant le type de résultat escompté, d'établir un tri (trier les actions dans des catégories prédéterminées), un classement (de la meilleure à la moins bonne), une description (décrire les solutions et leurs conséquences) ou un choix (choisir les bonnes solutions).
Dans les méthodes multicritères s'inscrivant dans cette voie, avant de construire ces relations de sur-classement, des seuils d'acceptation pour chaque critère peuvent être introduits, afin de cerner l'évolution des critères pris en compte.

L'avantage fondamental de ces méthodes se situe dans la richesse des relations possibles entre deux solutions (les conditions de sur-classement). De plus, elles n'imposent pas au décideur des contraintes de rationalité mathématique, telle que la transitivité de la préférence [TVERSKY 69] ou de l'indifférence. En effet, contrairement aux méthodes d'agrégation complète transitive, étant donné que les critères sont considérés séparément, ils peuvent être

[36] Appelée aussi méthode de sur-classement [VINCKE 89] ou encore approche de sur-classement de synthèse acceptant l'incomparabilité [MAYSTRE 94][ROY 96].

de nature très différente (puisqu'ils sont comparés deux à deux). Il est ainsi tout à fait possible de traiter simultanément des critères qualitatifs et quantitatifs, mesurables ou non mesurables. Les méthodes d'agrégation partielle, exclusivement développées dans le cas d'un nombre fini d'alternatives [ROY 93], sont très utilisées dans les problèmes comportant de nombreux aspects non quantitatifs et non commensurables, en l'occurrence ceux liés à l'environnement.

Cependant, outre le grand nombre de comparaisons[37] nécessaires entre les critères, le résultat de ces méthodes contient généralement plusieurs solutions plutôt qu'une seule, et dont certaines ne sont pas très intéressantes, mais suffisamment singulières qu'aucune solution ne les surclasse. Le choix de la relation de sur-classement joue ainsi un rôle important dans l'élimination des solutions non intéressantes et la détermination de la qualité des résultats.

Parmi les méthodes d'agrégation partielle, nous trouvons les méthodes des familles PROMETHEE et ELECTRE [ROY 68], l'optimum de Pareto, la méthode majoritaire [VANSNICK 86] qui peut être considérée comme l'ancêtre des méthodes de sur-classement et la règle de prudence qui part du fait que le décideur est généralement intéressé par le choix de la solution la moins vulnérable possible [ARROW 86].

II.3.2.3. Approche d'agrégation locale itérative

Alors que d'un côté les méthodes d'agrégation complète utilisent une fonction de synthèse pour comparer entre elles l'ensemble des solutions et de l'autre les méthodes d'agrégation partielle comparent directement deux à deux toutes les solutions, les méthodes d'agrégation locale[38] quant à elles, ne traitent simultanément qu'un sous-ensemble local de solutions (solutions trouvées localement). En plus, et contrairement à l'approche d'agrégation partielle qui suppose un nombre raisonnable de solutions possibles, les méthodes de cette approche s'utilisent en présence d'un grand nombre de solutions.

Ces méthodes, principalement développées dans le cadre de la programmation mathématique à objectifs multiples [WHITE 90], alternent les étapes de recherche des solutions et les étapes d'interaction avec les décideurs [ROY 93].

- *Etape de recherche/exploration* : A partir d'une solution de départ aussi bonne que possible, un processus itératif à base d'exploration locale de l'espace de recherche consiste à

[37] En général, pour n solutions, il faut effectuer (n-1) ! opérations de comparaisons pour chaque critère. Cette manière de procéder exige donc (n-1)!m opérations de comparaison avec m le nombre de critères.
[38] Appelée aussi approche de jugement local interactif avec itérations essais-erreur [ROY 96][MAYSTRE 94] ou encore méthode interactive [VINCKE 89].

rechercher les solutions relativement proches de la solution initiale[39] [SCHÄRLIG 85].

- *Etape d'interaction* : L'ensemble des solutions retenues est communiqué aux décideurs afin de collecter des informations concernant leurs préférences vis à vis de ces solutions. La solution préférée par le décideur est retenue pour constituer la solution de départ pour l'itération suivante.

Cependant, cette approche qui ne favorise pas la négociation entre décideurs, nécessite une grande disponibilité de ces derniers pour choisir la meilleure solution à l'issue de chaque itération. Les préférences des décideurs ne pouvant être exprimées explicitement, certains conflits peuvent apparaître à chaque itération lorsque le décideur est constitué par un groupe d'acteurs. Il est en plus difficile de définir les conditions d'arrêt du processus de recherche de solutions ou de garantir la satisfaction du décideur après un nombre déterminé d'itérations.

II.3.3. Synthèse des méthodes multicritères

Les méthodes multicritères accordent plus d'intérêt à l'interaction avec les décideurs et à la prise en compte de leurs préférences qu'à la satisfaction et la modélisation de leurs contraintes. L'aspect contraintes est souvent peu évoqué voir absent dans les démarches d'analyse multicritère et semble être la dernière préoccupation de l'homme d'étude. En effet, les contraintes restreignent l'espace de recherche et contraignent la qualité de la décision choisie. De ce fait, l'aspect contraintes ne peut être écarté de toute procédure de prise de décision, dont le succès, passe certainement par la satisfaction de l'ensemble des contraintes exigées par les décideurs. Dans le cas contraire, la décision prise risque d'être inefficace, impraticable ou mal accueillie par les différents décideurs.

II.4. Discussion : Quelle méthode utiliser ?

Le premier problème pratique qui se pose pour une application concrète concerne le choix de la méthode appropriée parmi les différentes alternatives possibles. Ce choix est d'autant plus difficile qu'il n'existe pas de règle générale pour estimer l'efficacité d'une méthode face à un problème donné. L'adaptabilité d'une méthode dépend avant tout des possibilités d'intégration des connaissances spécifiques au problème et de l'effort nécessaire pour le codage et la représentation de ses paramètres.

[39] Nous trouvons ici certains principes des méthodes de voisinage.

Dans notre cas, le choix de la méthode doit assurer un certain compromis issu de la contradiction entre le caractère NP-complet de notre problème (complexité qui croit exponentiellement avec la taille du problème) et la contrainte de temps imposant des impératifs de réponse en temps contraint. En effet, le besoin essentiel, dans notre problème, est de trouver rapidement la meilleure solution qui satisfait l'ensemble des contraintes (ce qui est une tâche complexe) ou du moins, celles les plus pertinentes. Nous disposons donc d'un aspect mixte d'optimisation et de satisfaction des contraintes ce qui place notre problème dans le cadre d'optimisation sous contraintes.

Etant donné la complexité du problème et le nombre élevé de contraintes exigées, notre problème fait partie de la classe des problèmes NP-difficiles et l'exploration de l'ensemble des solutions potentielles à l'aide d'une méthode exacte s'avère très coûteuse.

De la même manière, la recherche de la meilleure solution à l'aide d'une méthode approchée nécessite un codage particulier et une adaptation spécifique en raison de l'aspect sur-contraint du problème.

Pour résoudre notre problème, les méthodes complètes et approchées sont plutôt complémentaires que véritablement concurrentes et peuvent coopérer afin d'exploiter la puissance de recherche des méthodes approchées et d'expression et de prise en compte des contraintes des méthodes exactes.

III. Synthèse générale

La figure 10 récapitule l'interaction entre l'optimisation, la satisfaction des contraintes et l'aide à la décision ainsi que la complémentarité entre les techniques approchées, exactes et multicritères de résolution des problèmes.

Les décideurs expriment leurs besoins qui reflètent les objectifs à atteindre et les contraintes à respecter. En fonction de ces besoins, le problème peut basculer vers l'optimisation, l'affectation sous contraintes ou l'aide à la décision. Si la satisfaction des contraintes est primordiale, le problème est alors un problème de satisfaction de contraintes qui consiste généralement à construire progressivement une solution en se basant sur les techniques exactes. Dans le cas contraire (la tâche de satisfaction n'est pas primordiale), la nature mono ou multicritère (déduite des objectifs) du problème ainsi que la satisfaction des conditions d'optimisation déterminent deux cadres possibles de représentation du problème, à savoir l'optimisation ou l'aide à la décision. La complémentarité entre les trois disciplines fait que les différentes techniques, à savoir les méthodes approchées, exactes et multicritères, peuvent

être utilisées conjointement pour confronter les problèmes réels. En effet, malgré une divergence des appellations et des termes employés, les techniques utilisées dans la résolution des problèmes sont souvent issues des mêmes principes.

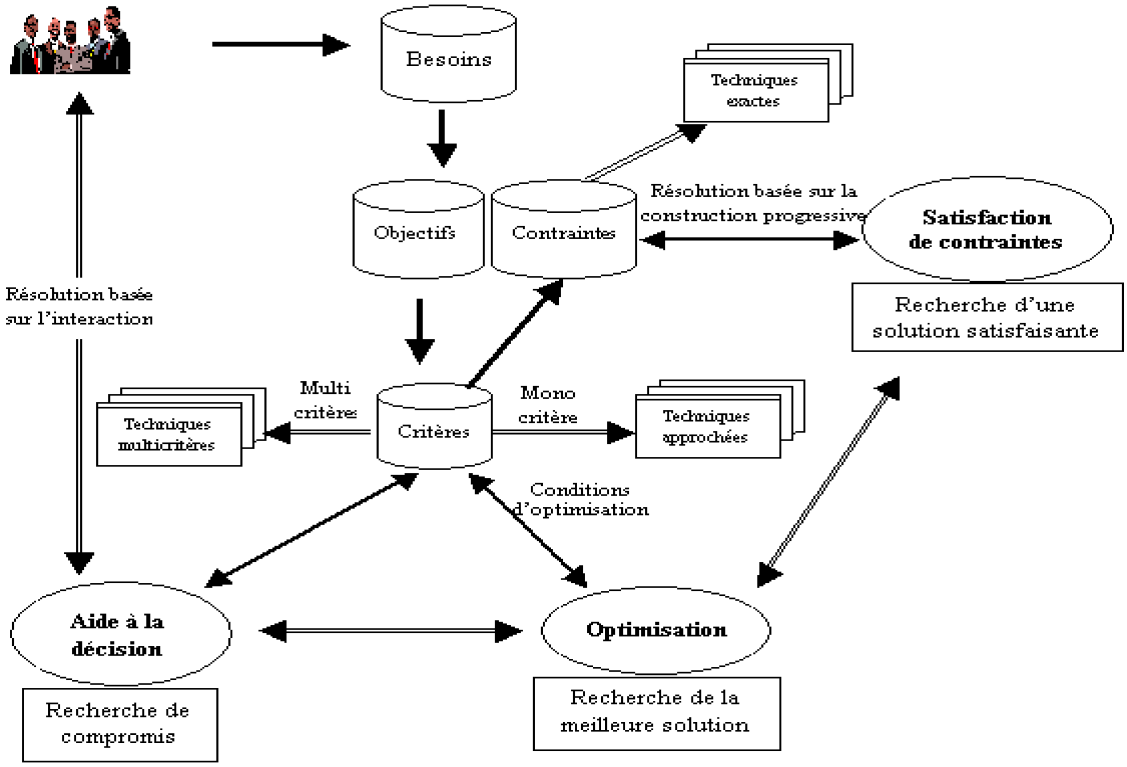


Figure 10 : Dépendance entre l'optimisation, la satisfaction des contraintes et l'aide à la décision et complémentarité entre les différentes méthodes de résolution.

L'étude de ces domaines de recherche permet de constater que l'aspect contraintes constitue un élément de réponse commun à ces trois disciplines et représente un aspect déterminant dans le choix des techniques applicables face à une situation donnée. Il contraint ainsi, de manière implicite ou explicite, la marge de manœuvre de l'homme d'étude et la qualité de la solution pouvant être proposée.

Cependant, malgré sa pertinence, il semble que l'aspect contraintes reste mal analysé et manque de modélisation et de fondements génériques de formulation et de traitement. L'intérêt reste souvent porté sur la démarche et technique de résolution plutôt que sur la modélisation du problème lui-même et il s'avère important d'accorder plus d'attention à cette tâche de modélisation.

C'est dans cette perspective, que nous espérons développer un cadre générique et interactif d'acquisition et de modélisation des objectifs et des contraintes pour les problèmes d'optimisation. Le cadre applicatif de ce travail vise à contribuer au développement de nouvelles méthodes pour aider les agriculteurs à mieux gérer leurs espaces individuels afin de contrôler et résoudre leurs problèmes collectifs.

CHAPITRE 3 :

Modélisation statique des objectifs et des contraintes

Sommaire

I. Orientations choisies

L'étude des domaines de l'optimisation, de l'affectation sous contraintes et de l'aide à la décision permet de constater la complémentarité entre ces trois domaines qui partagent de plus les mêmes techniques de résolution.
Cependant, dans ces trois disciplines, les chercheurs accordent plus d'importance à la résolution du problème qu'à sa formalisation. Les travaux menés jusqu'à présent se sont plus préoccupés du choix et du développement de la bonne technique permettant de satisfaire les contraintes plutôt que de modéliser les contraintes elles-mêmes. Ainsi, quelles que soient les approches, il s'avère important d'assurer une meilleure formalisation des contraintes dans le cadre d'une modélisation générique des différents aspects d'un problème.
L'importance de la modélisation apparaît notamment lors de la confrontation à des applications complexes ou faisant intervenir le facteur humain que ce soit dans la définition du problème, sa résolution ou la mise en œuvre des solutions proposées. Son impact sur la validation du système développé est ressenti déterminant lors du passage de l'aspect théorique (formel) à l'aspect pratique (applicatif). Dans les systèmes mal modélisés, la décision prise, comme étant optimale ou de bonne qualité au moment de l'étude du problème, risque d'être impraticable au moment de sa mise en œuvre sur un cas réel à cause du non-respect des exigences des décideurs.
En effet, quelle que soit la discipline dans laquelle s'inscrit le problème et qu'il s'agisse de la recherche de toutes les solutions, d'une bonne ou de la meilleure solution, les contraintes jouent un rôle fondamental en tant que clauses restrictives limitant la marge de manœuvre du concepteur et la qualité de la réponse pouvant être apportée à un problème donné. La satisfaction des contraintes s'impose comme étant un élément fondamental méritant plus d'intérêt et une meilleure modélisation par le concepteur afin de mieux définir et gérer la complexité du problème et de concevoir des systèmes plus robustes et plus représentatifs de la réalité.

C'est dans cette perspective, que se situe notre travail qui consiste à modéliser les contraintes du système afin d'apporter des éléments de réponse aux problèmes complexes ou sur-contraints. Nous cherchons ainsi à concevoir une démarche progressive dans la résolution d'un problème (figure 11), notamment en intégrant les décideurs dans un processus interactif et itératif de définition et de négociation des contraintes. La mise en œuvre de notre modélisation se fait à travers une démarche combinée qui intègre des phases d'identification

des besoins, d'optimisation, de satisfaction de contraintes et d'aide à la décision pour la gestion des problèmes collectifs entre des acteurs individuels.

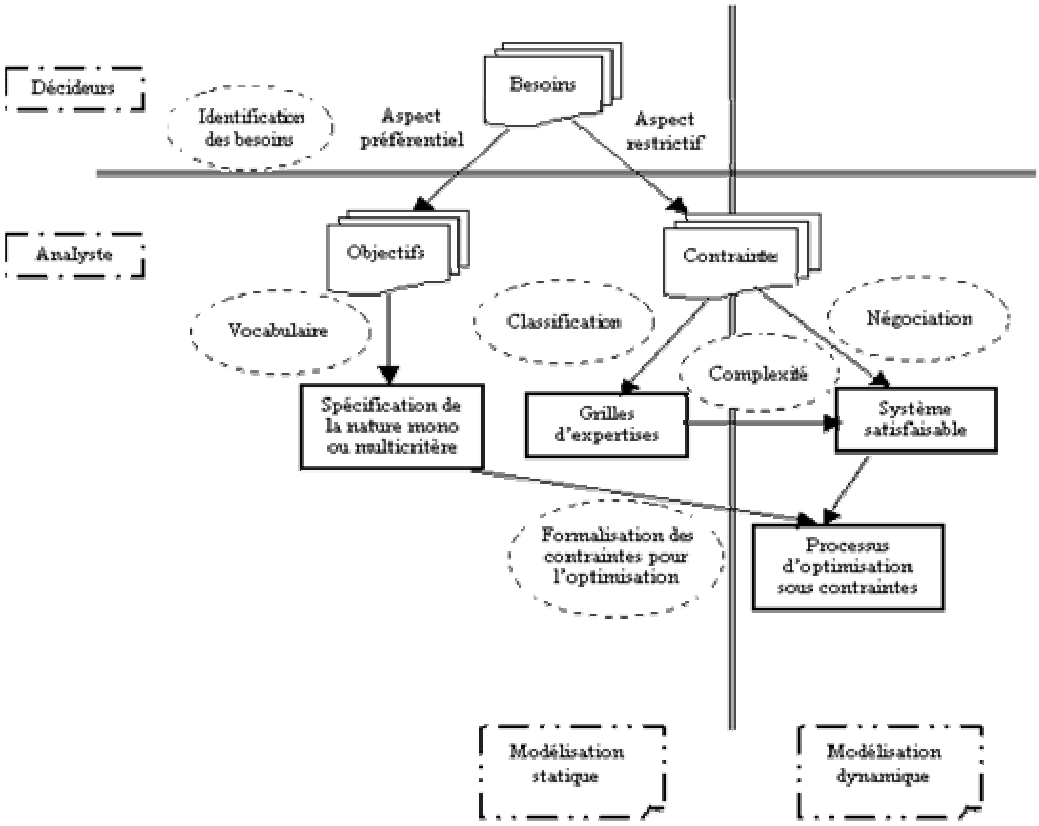


Figure 11 : Démarche progressive suivie pour la modélisation du problème.

Notre apport s'articule autour de deux axes principaux (figure 11) : un axe statique pour la définition et l'analyse des besoins des décideurs et un axe dynamique pour l'interaction et la résolution du problème.

L'axe statique, traité dans ce chapitre, cherche à amener les décideurs à définir leurs objectifs et leurs contraintes et à aider le concepteur dans l'analyse et la modélisation de ces besoins. Ces derniers sont définis via un vocabulaire d'échange d'informations entre les décideurs et le concepteur, permettant de déterminer la nature mono ou multicritère d'un problème et de déduire les techniques pouvant être appliquées face à la situation étudiée. Nous distinguons ainsi deux aspects. Un premier aspect orienté optimisation est lié à la définition, par le concepteur, des critères à optimiser traduisant les objectifs à améliorer et qui peuvent être déduits des besoins des décideurs exprimant des souhaits. Le deuxième aspect orienté satisfaction de contraintes, permet de modéliser les contraintes à respecter et qui peuvent être déduites des besoins des décideurs exprimant une forme restrictive ou exigeante. Les contraintes exigées devant être de complexité raisonnable afin d'accorder une certaine marge

de manœuvre au système, une classification des contraintes suivant différents paramètres est proposée et permet d'analyser la complexité des contraintes et les traitements nécessaires à leur satisfaction. A travers l'axe statique, nous pouvons constater que les décideurs expriment conjointement des besoins d'optimisation (objectifs) et de satisfaction (contraintes) et que leurs contraintes exigées nécessitent souvent des traitements complexes ce qui ne permet pas, dans certains cas, de trouver une solution satisfaisante. Un axe dynamique, sujet du chapitre 4, tente de remédier à ce problème en intégrant les décideurs dans la négociation de leurs contraintes afin de les sensibiliser aux difficultés rencontrées et propose un cadre d'optimisation sous contraintes pour assurer l'amélioration des objectifs des décideurs et le respect de leurs contraintes négociées. Nous distinguons ainsi un aspect d'aide à la décision traduisant l'interaction entre le système et les décideurs et un aspect d'optimisation sous contraintes. L'interaction est assurée via une négociation visant à examiner et à mettre à jour les contraintes décideurs (saisies via l'axe statique) afin de retenir un système de contraintes satisfiable. La phase d'optimisation sous contraintes combine l'optimisation des objectifs et la satisfaction du système de contraintes retenu par la phase de négociation afin d'assurer une conduite mixte (de la résolution) d'optimisation sous contraintes. Cette combinaison vise à assurer les deux tâches simultanément et à empêcher ainsi le processus d'optimisation d'influencer la résolution au détriment de la tâche de satisfaction.

II. Identification des besoins

Nous traitons dans ce chapitre l'aspect statique qui se limite à la saisie et l'analyse des besoins des décideurs pour préparer leur traitement. Nous faisons donc ici abstraction de la manière de gestion et de satisfaction de ces besoins dans un processus de simulation (sujet de l'axe dynamique). Nous pensons que la tâche de saisie et d'analyse de données dégage deux aspects différents : un aspect relatif à ce que les décideurs espèrent améliorer et qui reflète une recherche d'optimisation et un aspect relatif à la prise en compte des contraintes exigées par ces mêmes décideurs et qui reflète plutôt une recherche de satisfaction de contraintes. C'est autour de ces deux aspects que s'articule le présent chapitre visant à définir, en collaboration avec les décideurs, les critères à prendre en compte pour assurer les processus d'optimisation et de satisfaction des contraintes. La tâche d'optimisation que nous traitons s'intéresse particulièrement aux objectifs à améliorer et qui peuvent être déduits des besoins des décideurs exprimant des souhaits. Une démarche progressive permet d'exprimer les objectifs en terme d'aspects, de critères et d'attributs élaborant ainsi une hiérarchisation permettant de

déduire la nature mono ou multicritère d'un problème et les techniques d'optimisation pouvant être appliquées. La tâche de satisfaction de contraintes, quant à elle, porte sur l'analyse et la classification des contraintes à respecter et qui peuvent être déduites des besoins des décideurs exprimant une forme restrictive.

II.1. Processus d'interaction analyste/décideurs

Deux types d'acteurs interviennent dans les problèmes de décision : Les décideurs qui expriment leurs besoins et le concepteur qui modélise et satisfait ces besoins. Chaque type d'acteur peut être représenté par un groupe d'individus de pouvoirs d'expression différents. En effet, l'interaction et l'échange d'informations entre plusieurs acteurs de vocabulaires d'expression différents impliquent une propagation de l'information à travers trois niveaux de hiérarchie :

- Un niveau *informel/flou* relatif au niveau d'expression des décideurs : « ce qu'il est demandé de faire ».

- Un niveau *sémantique/formel* relatif au niveau d'expression du concepteur de l'application : « ce qu'il faut faire ». A ce niveau, le concepteur doit veiller à assurer une certaine neutralité[40] afin de traduire de façon objective les besoins des décideurs et ne pas influencer leurs choix. Le résultat du passage du niveau informel au formel peut donc être influencé par le concepteur dont la perception et le comportement conditionnent largement le bon déroulement du processus de décision.

- Un niveau *algorithmique/applicatif* relatif au niveau de simulation : « ce qui est possible/préférable de faire ». Nous constatons généralement à ce niveau une perte d'information due à la divergence entre ce qu'il faut faire (la réalité) et ce qu'il est possible de faire (le modèle) au regard des moyens que dispose le concepteur (temps, outils, contraintes). Ce problème est fréquemment rencontré lors de la conception et la mise en œuvre d'un outil informatique.

Ce passage à travers ces trois niveaux hiérarchiques peut entraîner des pertes d'informations et/ou modifier la sémantique des besoins. La fiabilité de la modélisation du problème et l'efficacité du processus de décision dépendent largement du bon déroulement de la phase d'acquisition et de propagation de l'information afin d'assurer une traduction fidèle

[40] En se référant à [ROY 75], Joerin [JOERIN 97] précise qu'une neutralité totale est une règle de conduite et non une condition et qu'elle reste impossible à atteindre.

de tous les aspects de la problématique. Compte tenu de ces exigences, et afin de mieux se positionner face à de telles conditions (divergence entre le monde sémantique et algorithmique, multitude d'acteurs intervenants), nous procédons à la définition d'un vocabulaire commun d'expression des besoins et d'échange de connaissances entre des acteurs différents.

II.2. Acquisition des besoins

La définition du vocabulaire de passage d'information entre les différents niveaux hiérarchiques vise à assurer la cohérence de la sémantique des termes utilisés et à préserver tous les aspects du problème pour une meilleure modélisation des informations afin de pouvoir les exploiter convenablement dans la résolution du problème. Nous excluons ici tout ce qui est aspect données et nous nous focalisons sur l'aspect échange et passage des informations entre le concepteur et l'utilisateur (décideur).

Dans le domaine de gestion du territoire, un des problèmes les plus difficiles est de prendre en compte et de modéliser les besoins des décideurs. Ces besoins qui traduisent les attentes des décideurs vis-à-vis d'un système de simulation ou d'aide à la décision, expriment les recherches d'amélioration et les bornes à ne pas dépasser. Ils sont exprimés de façon informelle (souvent verbalement) et non structurée. Nous distinguons les besoins sous forme préférentielle (les recherches d'amélioration) ou restrictive (les bornes d'amélioration) :

- Aspect *préférentiel* : Les expressions qui décrivent un aspect préférentiel sous forme de souhaits sans aucune forme de restriction reflètent pour le concepteur des objectifs à atteindre.
- Aspect *restrictif* : Les expressions décrivant la nécessité de respecter certaines restrictions sur les besoins d'amélioration, reflètent les contraintes à satisfaire par le concepteur. Les contraintes s'attachent à fixer des bornes (hautes ou basses) d'amélioration pour les critères à optimiser. Elles traduisent donc un esprit de limitation ou d'exigence qui restreint la marge de manœuvre du concepteur pour atteindre les objectifs.

Les besoins, suivant les aspects préférentiels ou restrictifs qu'ils reflètent, permettent d'orienter le problème soit vers l'optimisation (aspect préférentiel plus important), soit vers l'affectation sous contraintes (aspect restrictif très exigeant) ou l'aide à la décision (besoin d'assistance pour clarifier les besoins et effectuer certains choix).

Notons que les objectifs peuvent être transformés[41] sous forme de contraintes et inversement. Cette transformation peut être expliquée par des choix de résolution, par exemple le passage d'un problème de CSOP à un CSP équivalent passe par la transformation[42] de la fonction objectif sous forme de contraintes (voir chapitre 2).

III. Modélisation de l'optimisation

L'aspect restrictif des besoins, suivant le nombre et la complexité des contraintes qu'il exprime, détermine la nature sous ou sur-contraint du problème. L'aspect préférentiel, quant à lui, suivant le nombre de critères qu'il comporte, détermine la nature mono ou multicritère du problème. Nous nous intéressons dans cette partie à l'aspect préférentiel. Nous proposons cinq termes permettant de définir le vocabulaire d'échange d'informations: les besoins, les objectifs, les aspects, les critères et les attributs et de déduire ainsi la méthodologie de spécification de la nature mono ou multicritère du problème. Ce vocabulaire n'est pas limité aux problèmes d'optimisation et peut être étendu pour exprimer tout problème de décision. Nous nous focalisons toutefois sur la modélisation des besoins du point de vue de l'optimisation afin de déterminer les critères à optimiser et les méthodes appropriées pour la résolution.

III.1. Vocabulaire d'échange d'informations

Le niveau d'expression des besoins, qui représentent l'unité verbale informelle, est insuffisant pour dégager précisément les objectifs et les contraintes des décideurs. Parmi ces insuffisances, nous citons :

- *Des incompatibilités* : Certains besoins peuvent être contradictoires et irréalisables simultanément.

- *Des informations redondantes* exprimant une redondance totale : Le même besoin peut être exprimé différemment par un ensemble de décideurs.

- *Des imprécisions* : Les besoins sont souvent exprimés de façon imprécise.

- *La non-conformité avec la réalité pratique* : Les décideurs ne relèvent pas en général de connaissances profondes dans le domaine d'intervention du concepteur

[41] Une illustration pratique de cette transformation sera traitée au chapitre 5.
[42] Ce passage est délicat dans les situations où la limite à atteindre pour la fonction objectif ne peut être déterminée.

(informatique/mathématique) et expriment leurs besoins indépendamment de tout aspect conceptuel ou de mise en œuvre informatique. De ce fait, certains besoins exprimés par les décideurs peuvent s'avérer irréalisables pratiquement.

Afin de remédier aux insuffisances liées au niveau d'expression des besoins, l'analyste doit traduire formellement, en termes d'objectifs, les différents besoins préférentiels tout en gardant un comportement neutre et objectif. Dans la démarche que nous proposons, nous considérons qu'un besoin (sous forme préférentielle) se traduit par un seul objectif. Un besoin est donc forcément mono-objectif. Les mêmes besoins expriment un seul et unique objectif. Nous distinguons :

- Un problème mono-besoin (et donc mono-objectif) : Tous les décideurs expriment le même besoin (de la même façon ou de façons différentes).
- Un problème multi-besoins : Les décideurs n'expriment pas le même besoin ou expriment plusieurs besoins différents.

III.1.1. Les objectifs : Unité sémantique

Contrairement aux besoins qui sont souvent exprimés de manière imprécise, floue ou ambiguë, les objectifs sont des expressions sémantiques traduisant formellement dans un langage précis, des besoins différents et bien définis (ou plusieurs besoins redondants). Les objectifs doivent couvrir tous les besoins des décideurs et peuvent être atteints par une maximisation, une minimisation ou un maintien dans un état stable. Nous distinguons deux types de problèmes :

- Mono-objectif : Tous les besoins se traduisent en un même et unique objectif.
- Multi-objectifs : Plusieurs objectifs traduisent les besoins des décideurs.

Cependant, certaines insuffisances liées à la complexité et à la redondance des objectifs peuvent être ressenties à ce niveau sémantique :

- *Complexité ou dérivation* : Certains objectifs sont complexes et renferment plusieurs aspects différents, relatifs au même objectif. Ils doivent être dérivés en aspects élémentaires plus faciles à analyser et à atteindre. Un objectif doit ainsi être minimal, i.e., si nous éliminions un aspect de l'objectif, il ne sera plus le même. Nous distinguons donc les objectifs élémentaires non décomposables et les objectifs complexes (multi-aspects) pouvant être décomposés en aspects élémentaires :

- Objectif élémentaire (mono-aspect) : L'objectif renferme un seul aspect. Nous parlons ici de l'aspect réel/conceptuel indépendamment de la possibilité de l'exprimer pratiquement.
- Objectif complexe[43] (multi-aspects) : L'objectif renferme plusieurs aspects qualitatifs ou quantitatifs liés au sens de l'amélioration recherchée.

- *Informations redondantes* (redondance partielle) : Plusieurs objectifs peuvent renfermer le même aspect. Le passage au niveau aspect permet d'éliminer cette redondance et constitue le dernier nœud (feuille) de l'arbre sémantique (figure 13).

III.1.2. Les aspects : Unité sémantique élémentaire

Un aspect est une unité sémantique de mesure qui représente un objectif élémentaire ou une partie d'un ou de plusieurs objectifs complexes. Un aspect peut donc exprimer la totalité ou une partie d'un objectif mais ne peut en aucun cas représenter à lui seul deux objectifs différents (deux objectifs renferment au moins un aspect différent). A cet effet, un problème multi-objectifs est forcément multi-aspects.

Une insuffisance est relevée à ce niveau sémantique et concerne l'impossibilité de conversion numérique de certains aspects renfermés dans les objectifs (exemple : satisfaction morale des décideurs). Nous distinguons donc des aspects convertibles en unités de mesures et des aspects inconvertibles. Un aspect convertible, i.e., pouvant être exprimé numériquement, est traduit en critère de mesure. Dans le cas contraire, i.e., l'aspect est inconvertible faute d'expressions disponibles ou suite à des choix de résolution, l'information qu'il contient sera perdue lors de ce passage du réel au modèle de simulation. Les objectifs ne renfermant que des aspects inconvertibles sont donc perdus. En plus, deux objectifs différents peuvent être exprimés par le ou les mêmes critères (mais pas par les mêmes aspects) et un problème multi-objectifs (naturellement multicritère) peut ainsi se ramener pratiquement à un problème mono-critère.

Ce passage forcé du multi au mono-critère explique la dissimilitude entre le réel et le simulé et peut même entraîner, à cause de la perte d'informations (aspects ignorés), l'invalidité du système conçu ou l'inapplicabilité des solutions proposées. En général, il est

[43] Il ne faut pas confondre ici un objectif complexe et un objectif compliqué. Un objectif complexe peut être décomposé en plusieurs objectifs élémentaires pour pouvoir les satisfaire. Un objectif compliqué n'est pas forcément décomposable et il est généralement difficile à analyser et satisfaire.

difficile de conserver la conformité entre la réalité métier et la réalité de simulation, notamment dans les applications complexes faisant intervenir le facteur humain. Cette conformité n'est assurée que lorsque tous les aspects sont convertibles et que chaque aspect est traduit convenablement et judicieusement en un critère.

III.1.3. Les critères : Unité numérique

Un critère[44] est une unité numérique (mathématique) de mesure qui exprime un seul aspect convertible. Cependant, le même critère peut exprimer plusieurs aspects différents pour des raisons de dépendance entre les aspects ou pour des choix de résolution (faute d'expressions adéquates). Les critères représentent donc des expressions mathématiques capables de traduire numériquement les objectifs du problème et servent comme élément de décision pour l'évaluation des solutions potentielles. Pour être bien formalisés, les critères doivent être :

- *Précis et exhaustifs* : Les critères doivent traduire de façon précise tous les aspects quantitatifs et qualitatifs renfermés dans les objectifs afin de ne pas perdre d'information.

- *Indépendants* afin d'éviter les redondances : Les critères dépendants doivent, dans la mesure du possible, être regroupés en un seul critère.

Dans un contexte d'aide à la décision, et en plus des conditions déjà citées, les critères doivent être réduits et faciles à interpréter afin de ne pas dépasser les capacités cognitives de l'être humain [ROY 85]. Belton [BELTON 90] cite dans ce contexte que la recherche en psychologie a montré que le cerveau humain ne peut considérer simultanément qu'un nombre limité d'informations (de critères).
Le niveau de décomposition mono ou multicritère du problème détermine les techniques de résolution applicables face à une situation donnée.

III.1.4. Les attributs

Chaque critère est exprimé au moyen d'un ou plusieurs attributs dont les valeurs respectives déterminent l'évaluation de ce critère. Un attribut[45] correspond donc à une variable ou un paramètre intervenant dans l'évaluation du critère. Le même attribut peut

[44] Plusieurs définitions du terme critère ont été proposées dans [ROY 85] du point de vue de l'aide à la décision sans toutefois proposer une démarche de progression dans la définition des besoins, objectifs, aspects et critères.
[45] L'identification entre critère et attribut peut être vérifiée dans le cas où un attribut sert de base à un jugement de préférence [ROY 85].

intervenir dans l'évaluation de plusieurs critères différents. Dans ce rapport, nous parlerons indifféremment de critère ou de fonction-critère. Un critère c_i peut être de type :

- Critère mono-attribut : $c_i\ (a)$: La valeur d'une seule variable ou attribut *a* détermine l'évaluation du critère.

- Critère multi-attributs : $c_i\ (a_1, a_2, ..., a_n)$: La valeur du critère dépend des valeurs de plusieurs paramètres ou attributs a_j.

Dans le cas d'un problème mono-critère, il existe un seul optimum pour le critère considéré, selon lequel la meilleure solution peut être déterminée. Le processus d'optimisation cherche ainsi à déterminer les valeurs que doivent prendre les attributs exprimant le critère considéré afin de trouver l'optimum ou d'améliorer au mieux ce critère. Dans le cas d'un problème multicritère, la détermination de ces valeurs est plus délicate. En effet, il n'existe pas un optimum commun pour l'ensemble des critères pris en compte, i.e., chaque critère possède son propre optimum qui est, généralement, différent de celui des autres critères. A cet effet, deux critères, pouvant être exprimés par les mêmes attributs (mais avec des fonctions d'agrégation ou des coefficients de pondération différents), peuvent avoir un sens d'évolution différent. Les valeurs prises par les mêmes attributs n'ont donc pas le même effet d'amélioration sur les différents critères.

III.1.5. Illustration

Nous illustrons le passage d'information entre les différents termes du vocabulaire par l'exemple suivant : Les agriculteurs (les décideurs dans notre application), trop touchés par les processus de risque (ayant des parcelles dans des endroits sensibles du bassin versant), expriment le besoin d'améliorer le risque de ruissellement sans dégrader leurs gains financiers dans leurs exploitations respectives. Les agriculteurs dont les exploitations se trouvent en amont du bassin versant, cherchent avant tout à améliorer leurs productions et leurs gains financiers. D'autres souhaitent tout simplement être satisfaits des solutions proposées. Outre la minimisation des dégâts causés par le risque, les instances locales cherchent à respecter les orientations stratégiques européennes et à favoriser l'implantation des cultures anti-ruisselantes. La traduction de cet exemple[46] (figure 12) sous formes de besoins, objectifs, aspects, critères et attributs, prend la forme suivante :

[46] Cet exemple, bien qu'inspiré de notre cadre applicatif, ne représente pas la totalité du problème traité. Il sert simplement à illustrer les différents termes du vocabulaire.

Besoins :

1. Améliorer le risque (décideur agriculteur aval) ;
2. Ne pas dégrader les gains financiers (décideur agriculteur aval) ;
3. Améliorer la production (décideur agriculteur amont) ;
4. Améliorer les gains financiers (décideur agriculteur amont) ;
5. Satisfaire les agriculteurs (décideur agriculteur) ;
6. Minimiser les dégâts causés par le risque (instances locales) ;
7. Respecter les orientations stratégiques européennes (instances locales) : pour des raisons de simplicité, ce besoin ne sera pas pris en compte dans cet exemple ;
8. Favoriser les cultures anti-ruisselantes (instances locales).

Objectifs :

1. Améliorer le risque : exprime les besoins 1, 6 et 8 ;
2. Maintenir[47] les productions : exprime les besoins 2, 3 et 4 ;
3. Satisfaire les agriculteurs : exprime le besoin 5.

Aspects :

1. Améliorer le risque : deux aspects convertibles :
   - Risque local au niveau de chaque exploitation (aspect 1) ;
   - Risque global à l'exutoire du bassin versant (aspect 2).
2. Maintenir les productions : 1 aspect convertible (aspect 3) ;
3. Satisfaire les agriculteurs : comporte 2 aspects :
   - Aspect inconvertible : satisfaction morale (aspect 4 → perdu) ;
   - Aspect convertible : satisfaction en production (maintenir la production) : aspect qui apparaît déjà dans le 2ème objectif (aspect 3).

Critères :

1. Minimiser le risque local : Le risque local est calculé en fonction de la pente des parcelles, de leur surface en amont et des coefficients de ruissellement des cultures occupantes ;
2. Minimiser le risque global : Le risque global est fonction de la sensibilité des parcelles et des coefficients de ruissellement des cultures occupantes ;

[47] Nous exprimons la production par la surface couverte par un type de culture dans une exploitation. Les surfaces des exploitations étant limitées, nous ne pouvons améliorer toutes les productions simultanément. Nous considérons à cet effet que les besoins 3 et 4 : 'améliorer la production' sont irréalisables et se traduisent à un objectif plus réaliste : 'maintenir la production'.

3. Maintenir le seuil de production par culture et par exploitation : La production d'une culture est fonction des surfaces des parcelles contenant ce type de culture.

Attributs :

1. Pente des parcelles : parcelle.pente (critère 1) ;
2. Surface en amont des parcelles : parcelle.surfAmont (critère 1) ;
3. Coefficient de ruissellement des cultures : culture.coefRuissel (critères 1 et 2) ;
4. Sensibilité des parcelles : parcelle.sensibilité (critère 2) ;
5. Surface des parcelles : parcelle.surface (critère 3).

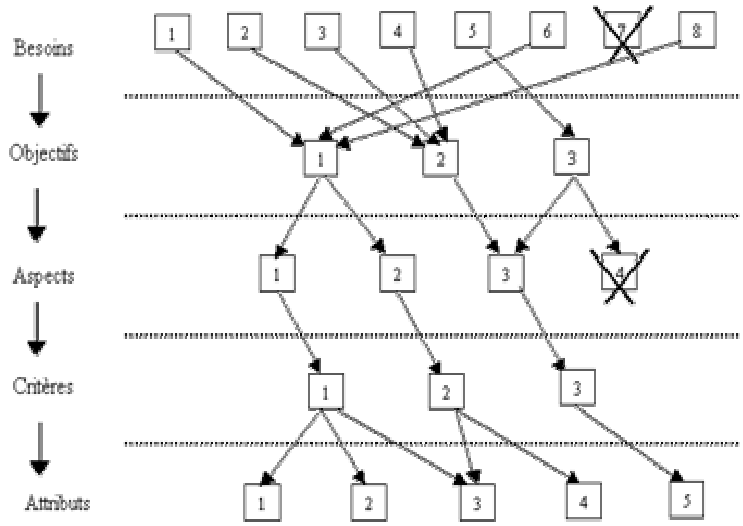


Figure 12 : Illustration de l'exemple en termes de besoins, objectifs, aspects, critères et attributs.

III.2. Méthodologie de spécification de la nature du problème

Les termes du vocabulaire permettent de déterminer la nature du problème étudié ainsi que les techniques pouvant être utilisées pour la résolution. Un problème d'optimisation (figure 13) peut être :

- *Mono-critère* : Cette classe de problèmes exprime les approches classiques d'optimisation où la fonction à optimiser peut naturellement être exprimée sous forme d'un unique critère de coût à minimiser ou de profit à maximiser au cours de la résolution. Un seul critère (fonction-critère) permet donc d'orienter le choix de la meilleure solution. Formellement, cette démarche s'exprime par :

Optimiser {V(s), s est une solution potentielle et V est la fonction d'évaluation des solutions}.

- *Multicritère* : Plusieurs critères indépendants traduisent le ou les objectifs à optimiser. D'une façon générale, pour améliorer un critère, il faut déterminer le jeu de paramètres

(attributs) permettant de vérifier les contraintes tout en améliorant au mieux ce critère. Dans le cas d'une optimisation multicritère, il s'agit plutôt de déterminer l'ensemble des solutions permettant d'optimiser au mieux les différents critères (ou un maximum de critères), sachant qu'une solution qui améliore un critère, n'améliore pas forcément un autre. Formellement, cette démarche s'exprime par :

Optimiser $\{Vc_1(s), Vc_2(s), \ldots, Vc_k(s)$, s est une solution potentielle, c_i est un critère à améliorer, Vc_i est la fonction d'évaluation des solutions pour le critère $c_i\}$.

Les techniques classiques d'optimisation s'attaquent généralement aux problèmes mono-critère. Toutefois, ces techniques peuvent être étendues aux cas multicritères en combinant l'ensemble des critères avec des heuristiques ou des contraintes supplémentaires dans le but de mieux contrôler l'évolution des critères et la qualité des solutions proposées. Dans le cas contraire, il faut plutôt faire appel à des techniques multicritères ayant pour objectif la recherche d'un bon compromis entre les différents critères.

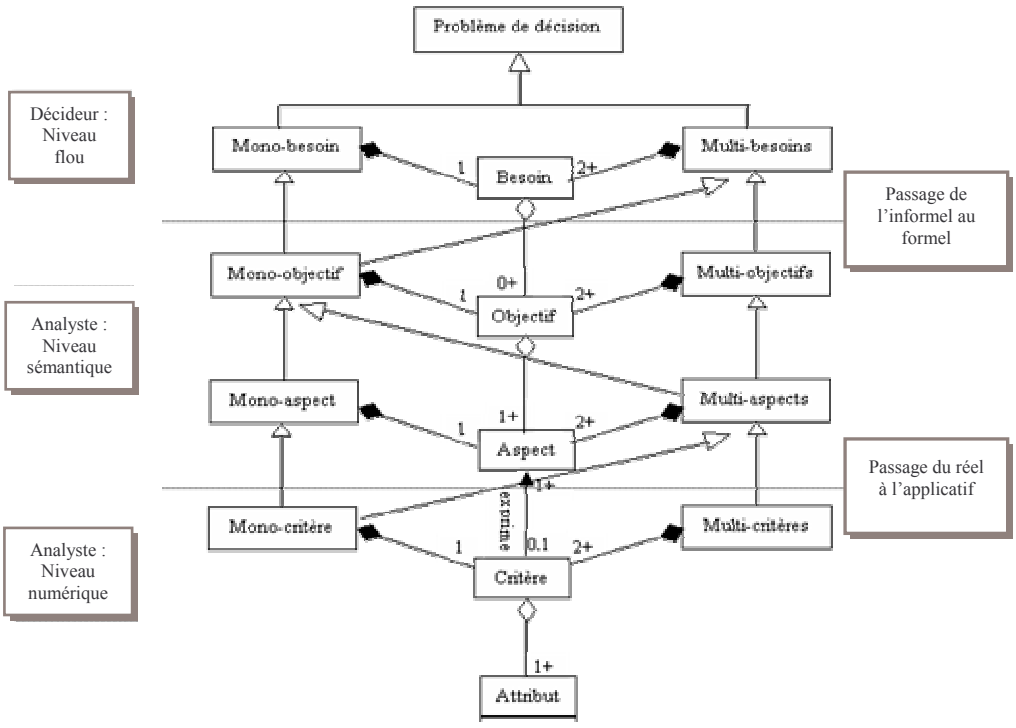


Figure 13 : Modélisation UML [BOOCH 99] du passage d'information entre les niveaux hiérarchiques.

La figure 13 modélise le passage des besoins entre les niveaux : informel/flou, formel/sémantique et numérique/applicatif. Un problème de décision cherche à satisfaire les besoins des décideurs exprimés sous forme d'objectifs et de contraintes. Nous distinguons généralement les problèmes mono-besoin où les décideurs expriment le même besoin et les

problèmes multi-besoins comportant plusieurs besoins différents. Le passage du niveau informel au formel traduit un problème mono-besoin en mono-objectif, i.e., le besoin se traduit en un seul objectif. Par contre, un problème multi-besoins peut se traduire à un problème mono ou multi-objectifs (plusieurs objectifs traduisent les besoins des décideurs). Un aspect étant un objectif élémentaire, un problème multi-objectifs est forcément multi-aspects. Chaque aspect doit naturellement être traduit en un critère numérique, ce qui n'est pas souvent le cas à cause de la dissimilitude entre le réel et l'applicatif. A cet effet, un problème multi-objectifs (naturellement multicritère) peut ainsi se ramener pratiquement à un problème mono-critère.

Le passage d'information contenue dans les besoins (niveau décideur) de l'informel au formel et du réel à l'applicatif permet de déduire les critères (niveau concepteur) pris en compte dans la résolution d'un problème mono ou multi-besoins.

IV. Modélisation des contraintes

Nous avons vu dans ce chapitre que les décideurs expriment leurs besoins sous forme préférentielle ou restrictive. La première forme a fait l'objet d'une analyse visant à déterminer la nature mono ou multicritère d'un problème. Etant dans un cadre mixte d'optimisation et de satisfaction des contraintes et afin de prendre en compte tous les aspects du problème, nous nous intéressons dans cette partie à la forme restrictive reflétant les contraintes à respecter. Les contraintes étant des conditions liant des connaissances[48], leur modélisation passe par la définition des connaissances qu'elles lient. Nous présentons tout d'abord la définition et la classification des connaissances suivant deux couches : la couche métier et simulation. Nous proposons ensuite diverses classifications des contraintes permettant d'analyser leur complexité et leur comportement au cours du temps, suivant différents niveaux d'expertises.

IV.1. Définition et classification des connaissances

Nous commençons cette partie par la définition des termes : donnée, information et connaissance dont l'utilisation révèle souvent certaines ambiguïtés et peut entraîner des confusions.

[48] Les contraintes peuvent aussi être considérées comme des connaissances.

- ***Donnée :***

Une donnée est une valeur pouvant exprimer un état, une mesure ou un choix et peut être représentée sous une forme cognitive, mathématique etc. (e.g., culture = "Blé"). Une autre définition proposée dans [PRINCE 96] considère la donnée comme étant un signifiant susceptible d'être capté, enregistré, transmis ou modifié. Selon G. Bateson [BATESON 77] : « Donnée ne veut pas dire événement ou objet mais, dans tous les cas, trace, description ou souvenir de certains évènements ou objets ».

- ***Information :***

V. Prince [PRINCE 96] définit l'information comme étant un signifié transporté par une donnée. Chaque donnée possède une ou plusieurs significations qui peuvent changer selon les personnes et/ou le contexte (une grande parcelle : une bonne parcelle, une parcelle à risque,…). Une information peut refléter la signification d'une donnée dans un contexte particulier. Elle peut être un savoir (une meilleure pratique agricole) ou un état (la surface d'une parcelle, l'état du sol).

- ***Connaissance :***

La connaissance est la faculté de connaître, de comprendre et de percevoir des données ou des informations. Elle exprime des relations entre des informations et permet de générer ou de déduire de nouvelles données conformément à un raisonnement. La connaissance peut être vue comme une procédure d'affectation du sens (information) à un signifiant (donnée). C'est donc ce qui est associé au passage entre le signifiant et le signifié [PRINCE 96][ALQUIER 90]. Exemple : La pente de la parcelle est grande, il est donc impossible d'utiliser les machines agricoles, il faut y placer de l'herbe.
L'ensemble des connaissances permettant d'analyser, de comprendre et d'entreprendre des actions, constitue le savoir-faire. Dans notre approche, nous distinguons les connaissances suivant deux couches : la couche métier et simulation (figure 14).

a. Connaissances métier

Ces connaissances sont définies par les experts et permettent de comprendre et analyser la problématique étudiée. Nous distinguons :

- *Les connaissances spécialisées* : Ces connaissances portent sur la discipline étudiée, dont les limites sont marquées avec précision. Exemple : Les connaissances agronomiques, géographiques etc.

- *Les connaissances évidentes* ou d'ordre général : Ces connaissances expriment des règles générales applicables à des disciplines différentes. Exemple : Les lois et les principes.

b. Connaissances de simulation

Ces connaissances sont nécessaires au concepteur afin de pouvoir résoudre les éventuels problèmes. Nous distinguons :

- *Les connaissances de modélisation* : Elles reflètent et traduisent les caractéristiques du système que nous voulons simuler.

- *Les connaissances de compétences* : Elles sont relatives au domaine de compétences du concepteur de l'application. Elles influencent son comportement face à une situation donnée : comment coder correctement et fidèlement le problème, gérer la complexité etc. Exemple : Les connaissances algorithmiques, linguistiques, etc.

- *Les connaissances relevant de l'expérience* : Elles sont issues de l'apprentissage du concepteur lors de la confrontation des applications diverses et permettent d'associer à une situation étudiée les actions appropriées.

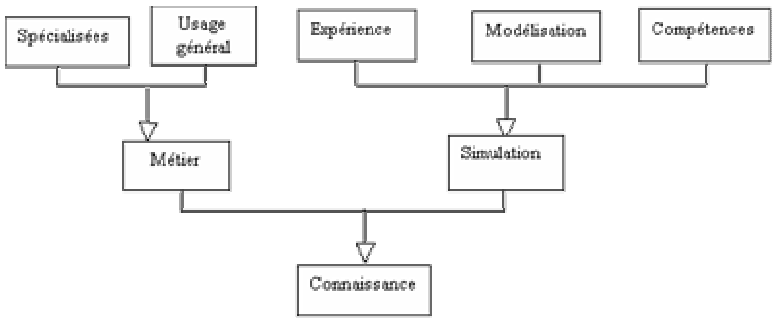


Figure 14 : Modélisation UML des connaissances dans un système de décision.

IV.2. Classification des contraintes

Une contrainte est une relation sémantique liant un ou plusieurs éléments du modèle conceptuel (instances, entités, …), et qui spécifie les conditions et les propositions qui doivent être respectées. La contrainte peut être décrite en langage naturel pour être compréhensible par les différents acteurs ou exprimée avec un langage formel permettant une meilleure modélisation de l'information. Les systèmes complexes intègrent souvent plusieurs acteurs qui interagissent et expriment des besoins et des exigences différentes. Il ressort de l'analyse de ces besoins que lors de la modélisation du problème, un ensemble de contraintes de

différentes natures doit être pris en compte selon des priorités plus ou moins importantes (les contraintes exigées n'ont pas la même importance pour les décideurs).

Les décideurs étant souvent de domaines de compétences différents, une tâche délicate consiste à établir des classifications des contraintes adaptées au domaine de compétences de chaque type d'acteur et à l'expression de ses besoins. Ceci permet de mieux modéliser les contraintes du système et de gérer leur complexité et leur préférence par rapport aux décideurs. Nous étudions dans cette partie différents paramètres de classification ainsi que leur contribution à l'élaboration de plusieurs grilles de contraintes visant des expertises différentes.

Généralement, la répartition des contraintes en différentes classes dépend de la problématique étudiée et du point de vue du concepteur de l'application. Dans la littérature, plusieurs critères ont été utilisés pour classer les contraintes, à savoir :

- *L'arité* des contraintes (voir chapitre 2 sur l'état de l'art) permet de distinguer les contraintes un-aires, binaires ou n-aires, tel est le cas dans les problèmes de satisfaction de contraintes [MACKWORTH 77].

- *L'incertitude* permet dans les CSPs flous [ROSENFELD 76] de distinguer les contraintes certaines et incertaines.

- *La priorité* permet de distinguer les contraintes dures qu'il faut absolument satisfaire, des contraintes souples qu'il est préférable de satisfaire. Ce paramètre de classification est utilisé par certaines extensions de CSP, en l'occurrence les CSPs possibilistes [SCHIEX 92].

- *Le domaine* des contraintes [TSANG 93] distingue les contraintes numériques (sur des entiers, des réels ou des rationnels), les contraintes symboliques (sur des domaines finis), et les contraintes booléennes (sur des types booléens). Un nouveau type nommé contrainte de type, de trait et d'arité (sur des structures à trait 'feature structure' qui étendent les termes structurés de la logique du premier ordre) a été défini par [CODOGNET 95].

Pour ce qui concerne notre contexte de simulation, nous définissons sept critères de classification :

- Le niveau d'acquisition des contraintes.

- La préférence sur les contraintes.

- La nature intrinsèque des contraintes.

- L'évolution dans le temps métier.
- Le comportement dans le temps de simulation.
- Le niveau d'abstraction des contraintes.
- L'arité conceptuelle (liens entre les éléments conceptuels).

Ces critères permettent de définir 3 grilles de classification de contraintes visant des niveaux d'expertises différentes : décideur, métier et conceptuelle et de déduire ainsi une grille combinée résultante du couplage entre la temporalité métier et de simulation.

IV.2.1. Expertise décideur

La première classification vise le niveau d'expertise décideur et permet de distinguer les contraintes suivant la vision de l'utilisateur du système.

- ***Le niveau d'acquisition des contraintes :***

Ce critère permet une décomposition systémique en couches suivant l'acteur intervenant (décideur, concepteur) et la tâche correspondante du système (interne ou externe). Nous définissons ainsi un cadre à quatre niveaux de contraintes :

- *Niveau métier :* Ce niveau renferme l'ensemble des contraintes relevant du domaine étudié et qui sont généralement définies par l'expert métier.
- *Niveau moteur de résolution :* Un ensemble de contraintes applicatives sont définies par le concepteur conformément à la stratégie de résolution adoptée.
- *Niveau décideur :* Ce niveau concerne les exigences des décideurs qui expriment leurs choix et leurs attentes vis-à-vis du système. Nous distinguons les contraintes implicites reflétant les objectifs pertinents à l'application (e.g., ne pas dégrader la valeur du risque de ruissellement) et les contraintes explicites (e.g., respecter le cycle d'assolement des parcelles) qui déterminent le degré de liberté dont dispose le concepteur.
- *Niveau interaction système-utilisateur :* Ce niveau concerne les contraintes sur les requêtes décideurs afin de déterminer celles qui peuvent être traitées par le système de simulation.

- ***La préférence sur les contraintes :***

Le décideur a souvent tendance à privilégier un sous-ensemble de contraintes conformément à ses intérêts et sa vision des choses. En effet, dans les applications réelles, les contraintes à

satisfaire n'ont pas généralement la même importance et renferment des aspects multiples. Les travaux autour des problèmes de satisfaction des contraintes distinguent notamment les contraintes dures qu'il faut absolument satisfaire, des contraintes souples qu'il est préférable de satisfaire (critère de priorité[49] évoqué ci-dessus). Dans notre contexte de gestion de risque, nous introduisons un troisième niveau de préférence. Nous considérons que les décideurs exigent à la fois des contraintes *dures* (niveau hiérarchique 1), qui traduisent des obligations et doivent impérativement être satisfaites, des contraintes *souples ou flexibles* (niveau hiérarchique 2), qui représentent des préférences et sont à satisfaire au mieux et des contraintes *secondaires* (niveau hiérarchique 3), dont l'insatisfaction contraint peu la qualité (pour l'utilisateur) de la solution proposée. A cet effet, nous définissons une hiérarchisation de contraintes selon 3 niveaux de préférence :

- *Niveau hiérarchique 1* : Correspond au niveau le plus prioritaire exprimant les contraintes dures.
- *Niveau hiérarchique 2* : Correspond à un niveau moins prioritaire exprimant les contraintes flexibles.
- *Niveau hiérarchique 3* : Correspond au niveau hiérarchique le plus faible exprimant les contraintes secondaires.

Cette classification offre une décomposition simple et représente une vue générique de la répartition des contraintes suivant le niveau de perception des décideurs. Cependant, elle reste pauvre en informations et pas assez détaillée pour permettre une modélisation fine des contraintes.

La définition et l'échange de connaissances (et de contraintes) entre les experts (niveau métier), les décideurs (niveau utilisateur) et le concepteur (niveau résolution) nécessite une redistribution plus fine des contraintes afin d'établir un vocabulaire commun compréhensible par les différents acteurs intervenants dans le processus de résolution. L'expertise métier semble mieux répondre à ces exigences et offre ainsi une classification plus appropriée des contraintes.

[49] Nous retrouvons les principes de certaines extensions de CSP, en l'occurrence les CSPs possibilistes (partie I.2.4 du chapitre 2).

IV.2.2. Expertise métier

Cette grille de classification vise le niveau d'expertise métier et permet d'établir un meilleur dialogue entre le concepteur et les décideurs. Cette décomposition se base sur la nature des contraintes et l'évolution dans le temps métier.

- ***La nature des contraintes :***

Elle dépend du domaine de variation des variables liées par une contrainte et permet d'établir une distinction entre :

- *Les contraintes spatiales* : Ce sont des contraintes qui expriment une dimension spatiale «où faire quoi» qui doit être vérifiée en limitant l'utilisation de l'espace. Une contrainte spatiale $\mathbf{c_s}$ est définie sur une ou plusieurs variables spatiales exprimant des notions de surface ou de distance. Une variable $\mathbf{x_s}$ est dite spatiale si son domaine de variation est spatial (prend des valeurs spatiales). Exemple : L'herbe doit être positionnée sur les parcelles situées à l'exutoire du bassin versant.

- *Les contraintes temporelles* : Elles concernent des contraintes assorties d'une modalité temporelle. Une contrainte temporelle $\mathbf{c_t}$ est définie sur une ou plusieurs variables temporelles. Une variable $\mathbf{x_t}$ est dite temporelle si son domaine de variation est temporel. Exemple : Certaines cultures ne doivent pas se succéder dans le cycle d'assolement des parcelles.

- *Les contraintes alphanumériques* : Elles concernent les contraintes exprimant des expressions mathématiques ou des variables alphanumériques. Une contrainte alphanumérique $\mathbf{c_\alpha}$ est définie sur une ou plusieurs variables numériques, booléennes ou symboliques. Une variable $\mathbf{x_\alpha}$ est dite alphanumérique si son domaine de variation est alphanumérique. Exemple : L'objectif de production du blé dans l'exploitation X doit dépasser 3 hectares.

- *Les contraintes dérivées* : Ce sont des contraintes combinées résultant du couplage de différents types de contraintes que nous venons d'évoquer (figure 15). Nous distinguons entre autres les contraintes spatio-temporelles, spatio-alphanumériques et tempo-alphanumériques.

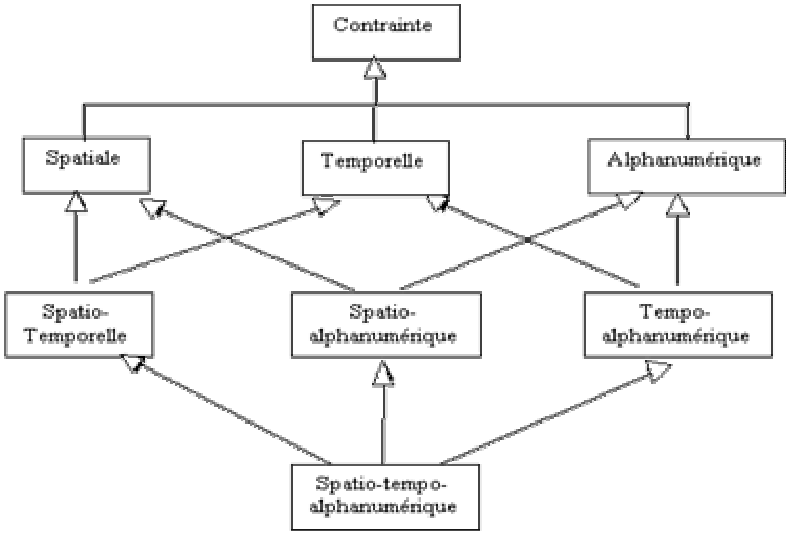


Figure 15 : Classement suivant la nature des contraintes.

- ***L'évolution dans le temps métier :***

Les processus de prise de décision concernant la gestion des risques naturels, et plus généralement le domaine de l'aménagement du territoire sont souvent lents et tout au long du déroulement du processus, des événements extérieurs peuvent intervenir. Ces événements qui ne font pas partie du processus modélisé, peuvent modifier les données du problème et influencer ainsi la prise de décision. Les connaissances sont donc susceptibles d'évoluer au cours du temps métier suite aux changements climatiques, à la situation du marché, au type de culture (culture d'hiver ou de printemps), à la période de l'année, etc. Exemple : Le type de sol n'a pas le même comportement en printemps et en hiver vis à vis de l'absorption de l'eau.

Une contrainte étant une condition qui doit être respectée pour une ou plusieurs connaissances, l'évolution de ces dernières au cours du temps entraîne une évolution des contraintes correspondantes. Il s'ensuit qu'une contrainte peut changer de comportement et/ou de signature (changement de paramètres). Autrement dit, une contrainte peut avoir plusieurs instances au cours du temps métier. La signature d'une contrainte correspond à l'ensemble de ses variables et des opérateurs les liant. Le comportement d'une contrainte correspond à sa sémantique et sa façon d'intervenir (le sens de la contrainte).
Cette idée trouve sa justification dans le fait qu'une solution ou décision prise par un système donné, peut être appliquée dans des conditions qui sont souvent différentes, totalement ou en partie, de celles qui ont permis de la découvrir au moment de la résolution [COURBON 93]. Dans ce sens, nous risquons de trouver impraticable (au moment de l'exécution), une décision

prise comme étant optimale au moment de la résolution. Cette évolution des connaissances entre le moment de la définition et la modélisation du problème et le moment de la mise en exécution des solutions développées (au cours de la modélisation) implique la nécessité de prendre en compte des paramètres supplémentaires, afin de prévoir ces changements et de distinguer ainsi les contraintes évolutives de celles qui ne le sont pas. Les solutions proposées doivent être assez souples pour pouvoir s'adapter à d'éventuels changements des données du problème, même si certaines évolutions restent difficiles à prévoir. Ceci nous amène à distinguer :

- *Les contraintes stables* : Ce sont des contraintes figées et leur statut est invariant dans l'intervalle de temps métier. Ceci impose la même signature et le même comportement au cours du temps.

- *Les contraintes évolutives* : Ce sont des contraintes qui peuvent évoluer au cours du temps métier en changeant de comportement et/ou de signature (changement de ses variables ou des opérateurs appliqués à ces variables).

Cette classification offre un vocabulaire compréhensible par les différents acteurs et renseigne sur l'évolution des contraintes à l'échelle métier que nous assimilons au cycle d'assolement. Elle reste cependant limitée au temps métier et ne permet pas de modéliser la complexité des contraintes et leur évolution à l'échelle du temps de simulation. Ce dernier point sera pris en compte par l'expertise conceptuelle.

IV.2.3. Expertise conceptuelle/de simulation

Cette expertise vise une décomposition informatique plus fine permettant d'analyser et de gérer la complexité des contraintes et leur évolution dans le temps de simulation. Trois paramètres sont ainsi définis : le comportement des contraintes au cours du temps de simulation, leur niveau d'abstraction et les liens conceptuels.

- ***Le comportement dans le temps de simulation :***

Ce critère exprime la possibilité de relaxation d'une contrainte au cours du temps de simulation. Nous considérons qu'une contrainte peut être relaxée par l'acceptation de valeurs (de variables) qui ne satisfont pas la contrainte, par la modification de sa condition de satisfaction[50] ou encore en autorisant l'instanciation des variables liées par des valeurs

[50] Exemple : La production en blé doit être supérieure à 3 Hectares. La contrainte peut être relaxée en modifiant sa condition de satisfaction : 'être supérieure à 3 Hectares' par 'être supérieure à 2 Hectares'.

n'appartenant pas à leur domaine[51]. Une contrainte peut être relaxée quel que soit son niveau de préférence (dure, flexible ou secondaire)[52]. La relaxation permet de distinguer les contraintes statiques et dynamiques :

- *Les contraintes statiques* : Ce sont des contraintes figées sur la durée d'une simulation. Elles ne peuvent être relaxées sous aucun prétexte et quelle que soit la stratégie de résolution et les facteurs externes et internes qui peuvent intervenir. Les contraintes statiques sont en majorité déclaratives et représentent la structure interne et les relations entre les entités manipulées.

- *Les contraintes dynamiques* : Ce sont des contraintes qui peuvent être relaxées au cours d'une simulation. Elles concernent des processus évolutifs et définissent souvent les règles qui doivent être respectées lors des opérations de manipulation ou de mise à jour des variables du problème (affectation de valeurs à ces variables etc.), afin de maintenir un état stable pour le système. Contrairement aux contraintes statiques, nous acceptons la relaxation de ces contraintes suite à :

  - Des raisons internes au système : liées à l'environnement de simulation.

  - Des raisons externes au système : liées à l'interaction avec le décideur.

- ***Le niveau d'abstraction :***

Ce critère conduit à déterminer deux types de contraintes :

- *Les contraintes simples* (classiques) : elles expriment des relations sémantiques entre les données du problème.

- *Les méta-contraintes* (niveau méta) : elles concernent des contraintes sur la structure d'autres contraintes.

Alors que le traitement des contraintes simples dans le processus de satisfaction a pour but d'attribuer une valeur aux variables du problème, la prise en compte de méta-contraintes construit ou ajuste la structure du problème lui-même, permettant ainsi sa reconfiguration au cours de la résolution.

[51] Ceci revient à élargir/modifier les domaines des variables liées par la contrainte.
[52] Nous distinguons deux couches : décideur et système. La préférence sur les contraintes est un paramètre décideur. Par contre, la relaxation est un paramètre système qui s'applique sur toutes les contraintes, indépendamment de leur niveau de préférence, afin de résoudre les problèmes sur-contraints.

- ***Les liens conceptuels/arité conceptuelle :***

Ce critère établit une distinction entre les contraintes et leurs interventions au niveau des entités et des instances conformément au modèle conceptuel de données. Ce paramètre permet de déduire la complexité conceptuelle des contraintes.

- *Contraintes sur les entités* : Le nombre d'entités impliquées dans une contrainte détermine sa complexité, en évaluant le nombre d'opérations d'indexation et de sélection nécessaires pour la vérifier. En effet, pour chaque entité impliquée dont la clé n'est pas donnée, la complexité augmente avec le nombre d'instances de l'entité concernée. Nous distinguons :

  - *Contraintes intra-entité* : concernent des données à l'intérieur d'une même entité.

  - *Contraintes inter-entités* : expriment une relation liant des données de plusieurs entités différentes.

- *Contraintes sur les instances* : Nous parlons ici de la nature conceptuelle de la contrainte (indépendante du temps) et non du nombre effectif d'instances impliquées dans la contrainte à un instant donné. L'évaluation de la satisfaction d'une contrainte dépend du nombre d'instances qu'elle implique simultanément. Nous distinguons :

  - *Contraintes mono-instance* : La contrainte peut être vérifiée pour une seule instance à la fois (indépendamment des autres).

  - *Contraintes multi-instances* : La vérification de la contrainte met en relation plusieurs instances simultanément.

Il faut distinguer ici les contraintes dont l'évaluation de leur satisfaction fait intervenir plusieurs instances à la fois (contraintes multi-instances) de celles faisant intervenir plusieurs instances différentes mais qu'une seule instance est vérifiée à la fois (contraintes mono-instance).

Exemple 1 : La somme des surfaces des parcelles contenant du blé doit être supérieure à un certain seuil. La satisfaction de la contrainte nécessite la vérification simultanée de toutes les parcelles contenant le blé. Cette contrainte est donc multi-instances bien qu'il puisse y avoir une seule instance (une seule parcelle contenant le blé) impliquée dans cette contrainte à un instant donné.

Exemple 2 : Les parcelles appartenant à monsieur X doivent contenir une culture fourragère. L'évaluation de la satisfaction de la contrainte fait intervenir plusieurs

parcelles/instances (toutes les parcelles de monsieur X), mais une seule instance est vérifiée à la fois (vérifier si chaque parcelle appartenant à monsieur X, seule, contient une culture fourragère). Cette contrainte est donc mono-instance.

Cette grille détaille davantage les contraintes et leur évolution dans le temps de simulation ainsi que leur complexité conceptuelle (suivant le paramètre arité conceptuelle). Cependant, l'analyse de la modalité temporelle reste limitée à l'échelle de la simulation et nécessite une projection à l'échelle du temps métier ce qui nous amène à définir une grille mixte combinant le temps métier et de simulation.

IV.2.4. Grille de correspondance métier/simulation

Les grilles métier et de simulation analysent les contraintes selon deux échelles de temps différentes : le temps métier et le temps de simulation. Pour mieux définir la modalité temporelle dans notre système, il s'avère important de combiner ces deux échelles de temps. A cet effet, nous procédons à une classification multi-échelles de temps pour mieux définir la notion de temporalité et le transfert d'échelles entre la réalité du terrain (le métier) et les exigences applicatives de simulation. L'échelle de temps adoptée est celle du temps réel qui intègre le temps métier et de simulation (figure 16).

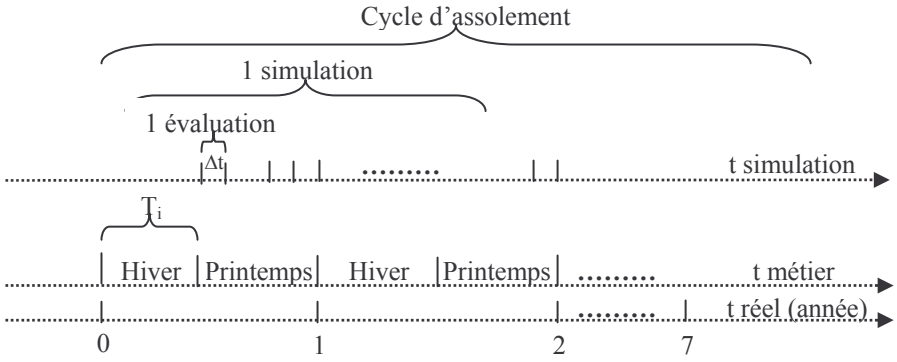


Figure 16 : Le transfert d'échelles entre la temporalité métier et de simulation.

Nous désignons par T_i la période dans le temps métier durant laquelle les connaissances métier ne changent pas. Une simulation est composée de plusieurs évaluations et peut s'effectuer durant plusieurs périodes métier T_i. Nous désignons par Δt le temps nécessaire pour effectuer une évaluation. Plusieurs évaluations peuvent avoir lieu dans la même période métier. Au cours d'une évaluation, la seule opération autorisée sur une contrainte est sa

relaxation. Au cours d'une simulation, une contrainte peut changer de comportement et/ou de signature et peut aussi être relaxée. Ceci nous amène à distinguer :

- *Contraintes statiques* : Ces contraintes ne sont pas relaxables au cours du temps de la simulation. Néanmoins, elles peuvent évoluer entre deux intervalles métier différents (T_i et T_{i+1}) en changeant de comportement et/ou de signature. Nous distinguons :

  - Les contraintes statiques stables : Ce sont des contraintes non relaxables, conservant le même comportement et signature.

  - Les contraintes statiques évolutives : Ce sont des contraintes non relaxables, pouvant changer de comportement et/ou de signature.

- *Contraintes dynamiques* : Ce sont des contraintes qui peuvent être relaxées, soit par le système, soit par l'utilisateur, tout au long de la résolution (dans Δt). Nous distinguons :

  - Les contraintes dynamiques stables : Ce sont des contraintes relaxables, ayant le même comportement et la même signature.

  - Les contraintes dynamiques évolutives : Ce sont des contraintes relaxables, pouvant avoir un comportement différent et/ou une signature différente.

Cette décomposition analyse donc les modalités temporelles au niveau du temps réel, et permet d'établir le tableau de classification suivant (figure 17) :

			Entité / Instance			
			Mono-entité		Multi-entités	
Temps Simulation	Relaxation	Temps métier	Intra-instance	Inter-instances	Intra-instance	Inter-instances
Statiques	Non relaxables	Stables				
		Evolutives				
Dynamiques	Système	Stables				
		Evolutives				
	Utilisateur	Stables				
		Evolutives				

Figure 17 : Classification multi-échelles offerte par la grille de correspondance métier/simulation.

IV.3. Synthèse de la modélisation des contraintes

Le système de contraintes se compose d'une ou de plusieurs contraintes, ayant chacune un niveau hiérarchique de préférence, et d'opérateurs (un-aires, binaires, …) agissant sur les

contraintes. Une contrainte est une condition appliquée sur une ou plusieurs connaissances et se compose d'un ensemble de variables liées par des opérateurs. Une variable peut être de type spatial (de surface ou de distance), temporel ou alphanumérique (booléenne, numérique ou symbolique). La figure 18 récapitule les liens conceptuels entre les connaissances, les contraintes et leurs différents composants évoqués tout au long de cette partie.

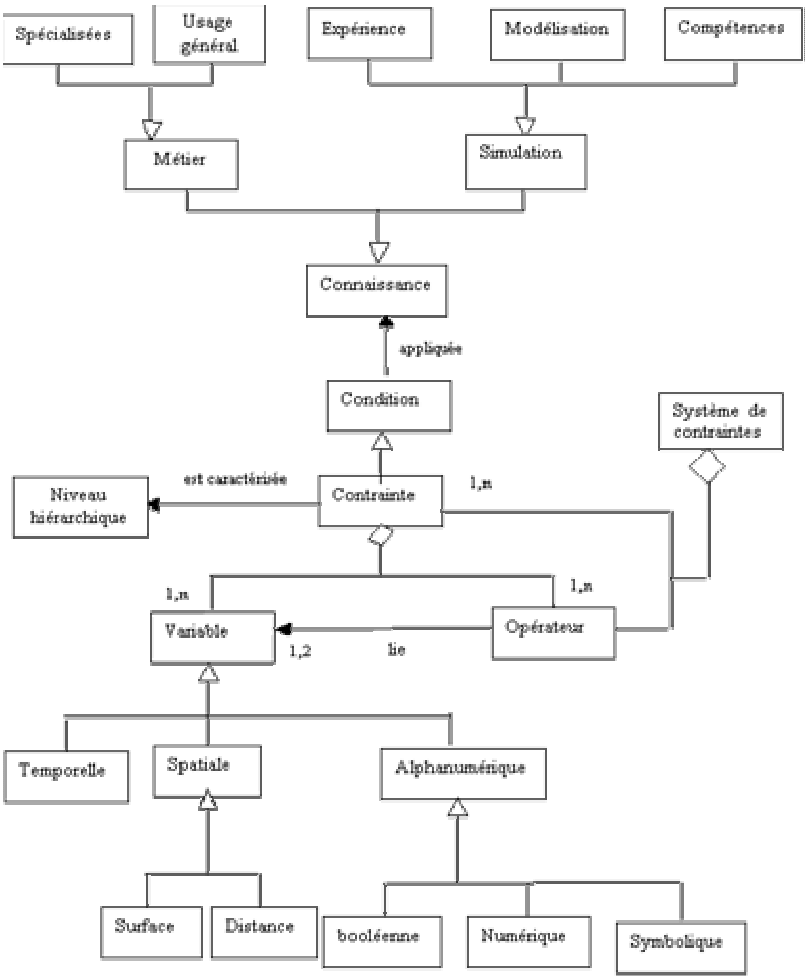


Figure 18 : Modélisation UML des relations conceptuelles entre les composants des connaissances et des contraintes.

V. Synthèse de la modélisation statique

Nous avons traité dans ce chapitre une modélisation statique offrant un mécanisme de définition et de formalisation des critères à optimiser conformément au vocabulaire d'échange d'information entre des acteurs différents. Cette étape permet de déterminer la nature mono ou multicritère du problème d'optimisation et de déduire ainsi les techniques appropriées pour la résolution. Le classement des contraintes suivant divers paramètres de classification permet

d'analyser et de gérer la complexité des contraintes et leur comportement au cours du temps métier et de simulation.

Un processus d'optimisation trouve généralement un sens plus évident dans les problèmes offrant un large choix de solutions. Dans un tel cas, le processus de résolution consiste à parcourir l'espace de solutions afin de retenir la meilleure. Cependant, dans les problèmes d'aménagement de territoire, les décideurs expriment naturellement beaucoup de contraintes et les problèmes sont souvent sur-contraints. Le système de contraintes correspondant ne permet pas de trouver une solution satisfaisant l'ensemble des contraintes exigées par les décideurs. Une tâche délicate consiste à pouvoir gérer ces contraintes et les négocier afin de retenir, en accord avec les décideurs, un sous-ensemble de contraintes de complexité raisonnable offrant ainsi un certain choix de solutions. En adoptant une telle approche, nous espérons aider les décideurs à exprimer et à mettre à jour eux-mêmes leurs contraintes plutôt que de décider à leur place. Ce point constitue le sujet du prochain chapitre qui propose également un cadre de formalisation et d'orientation de la résolution pour prendre en compte la complexité des contraintes et assurer leur satisfaction au cours d'un processus d'optimisation.

CHAPITRE 4 :

Modélisation dynamique

Sommaire

Nous avons vu dans le chapitre précédent l'aspect statique de définition et de modélisation des besoins des décideurs sous forme d'objectifs et de contraintes. Les décideurs étant impliqués dans la définition de contraintes souvent très complexes ou difficiles à satisfaire simultanément, nous souhaitons les intégrer dans le processus de résolution du problème et les sensibiliser aux difficultés rencontrées par le concepteur lors de la prise en compte de leurs contraintes.

A cet effet, nous traitons dans ce chapitre un aspect dynamique suivant deux axes. L'axe d'aide à la décision vise à intégrer les décideurs dans la saisie et la mise à jour de leurs contraintes et l'élaboration d'un ordre de satisfaction de ces contraintes. Cette intégration est assurée via une phase de négociation entre le système et les décideurs pour retenir un système de contraintes satisfiable. L'axe d'optimisation sous contraintes associe la tâche d'optimisation des objectifs définis par le vocabulaire d'échange d'informations (chapitre 3) et la tâche de satisfaction des contraintes retenues par la phase de négociation.

I. Modélisation de l'interaction

Nous cherchons à modéliser le processus de traitement et de gestion des contraintes en collaboration avec les décideurs. A cet effet, nous définissons un mécanisme (langage) de génération de plan de satisfaction des contraintes à partir des besoins (restrictifs) exigés par les décideurs. Le plan de satisfaction, qui représente la liste des contraintes ordonnées suivant leur pertinence par rapport au système, permet d'élaborer le graphe opérationnel (dépendant de l'application) au moyen d'opérations et de scénarios sur le système de contraintes. Ce graphe opérationnel de traitement des contraintes, retenu en accord entre le système et les décideurs, représente les contraintes effectives à prendre en compte dans le processus de résolution.

Le système de simulation cherche ensuite à favoriser la satisfaction des contraintes les plus pertinentes à l'application en suivant le graphe opérationnel. La pertinence système des contraintes est déterminée en fonction de trois paramètres (métier, décideur et système) afin d'intégrer et de prendre en compte tous les aspects du problème. L'élaboration du graphe opérationnel de traitement des contraintes passe par trois phases : une phase de saisie des besoins des décideurs, une phase de classement et d'organisation de ces besoins suivant leur

pertinence système et une phase de négociation entre le système et l'utilisateur pour retenir un système de contraintes satisfiable[53].

I.1. Saisie des besoins

Nous accordons aux décideurs la possibilité d'exprimer directement leurs choix et exigences vis-à-vis du système de simulation suivant une syntaxe bien définie. Ceci est réalisable formellement au moyen d'un langage défini à l'aide d'une grammaire générative. Les interfaces homme/machine ont fait l'objet de nombreux travaux sur ce sujet [STROMBONI 85]. Nous nous positionnons ici au niveau conceptuel/formel. A cet effet, nous définissons une grammaire conformément à la notation BNF (Forme de Backus-Naur) permettant une représentation finie du langage de saisie et de manipulation de la liste des contraintes à négocier avec les décideurs. La vérification de la syntaxe des besoins saisis par les décideurs passe par une analyse lexicale et syntaxique[54] afin de vérifier leur conformité au regard du langage spécifié par la grammaire. L'analyse lexicale permet de traduire un programme source en une chaîne de symboles qui constitue le langage d'entrée de l'analyseur syntaxique. L'analyse syntaxique quant à elle, permet de déterminer si une chaîne de symboles (les unités lexicales) peut être engendrée par une grammaire et de vérifier ainsi si l'utilisateur a respecté la grammaire du langage source lors de la saisie.

Nous définissons un exemple issu de la grammaire définie dans le cadre de notre travail et nous renvoyons à l'annexe 2 pour de plus amples détails sur la structure complète de cette grammaire. L'intervention d'un acteur consiste en une expression de ses besoins suivie d'une suite_intervention. Une suite_intervention prend la forme d'un séparateur et d'un scénario sur la liste des contraintes. Elle peut éventuellement être vide. Le scénario ne sera pas traité dans cet exemple pour des raisons de simplicité. L'expression des besoins commence avec le label 'Besoin' suivi de deux points et d'une liste de contraintes à satisfaire. Une liste de contraintes est une contrainte suivie d'une autre liste. Elle peut éventuellement être une liste vide. Une contrainte prend la forme d'un triplet : attribut, comparateur, valeur. Un attribut est une suite alphanumérique (chaîne de caractères) contenant une lettre et éventuellement une autre chaîne de caractères.

[53] Il est difficile de déterminer la satisfiabilité d'un système, notamment pour les applications complexes. Nous visons par le terme 'satisfiable', un système de contraintes assez souple offrant une certaine marge de manœuvre au concepteur.

[54] Il existe également une troisième phase d'analyse sémantique qui consiste à vérifier le sens (la sémantique) du texte entré par l'utilisateur.

I.1.1. Définition de la grammaire

Une grammaire formelle est un quadruplet G = (*X,V,S,P*) où :

- *X*, vocabulaire, est un ensemble fini de symboles appelés terminaux.
- *V*, disjoint de *X*, un alphabet non terminal composé de l'ensemble de symboles non terminaux ou variables.
- *S*, élément de V, appelé axiome ou start.
- *P*, un ensemble de règles de production : $P = \{(u, v)/ u \rightarrow v, u \in (X \cup V)^+ \text{ et } v \in (X \cup V)^*\}$.

Une grammaire génère les mots d'un langage en dérivant l'axiome. Un langage généré par une grammaire G est l'ensemble des symboles terminaux qui peuvent être dérivés en une ou plusieurs étapes du symbole de départ. Chaque grammaire génère un seul langage. Un langage peut être généré par plusieurs grammaires.

I.1.2. Notation BNF

La forme BNF simplifie l'écriture des grammaires en regroupant plusieurs règles en une seule règle BNF, selon un format spécifique. Nous utilisons dans l'illustration de l'exemple les notations suivantes :

- Les symboles terminaux sont écrits entre ‘ et ’ ;
- Les symboles non terminaux sont écrits entre ‘<’ et ‘>’ ;
- La ‘→’ de dérivation est remplacée par ‘::=’ ;
- Le symbole ‘|’ sert à exprimer des alternatives.
- Les répétitions 0, 1 ou plusieurs fois : T ::= { h } signifie : $T \rightarrow \varepsilon$; $T \rightarrow hT$;
- L'option (répétitions 0 ou 1 fois) : T ::= [h] signifie : $T \rightarrow \varepsilon$; $T \rightarrow h$;
- Les répétitions 1 ou plusieurs fois : T ::= (h)+ signifie : $T \rightarrow h$; $T \rightarrow hT$.

La traduction de l'exemple ci-dessus cité en notation BNF prend la forme suivante :

```
X = {'a'..'z','A'..'Z','0'..'9','<','>','<>','<=','>=','=',':','&',
'Besoin'} ;
```

```
V = {Intervention_acteur, Expression_besoin, Suite_intervention,
Scénario_liste, Séparateur, Label_besoin, Deux_points, Liste_contraintes,
Contrainte, Comparateur, Attribut, Chaîne, Valeur, Lettre} ;
```

```
S = {Intervention_acteur}.
```

En ce qui concerne l'ensemble P des règles de dérivation, nous distinguons les règles générant des variables et celles générant des terminaux (tokens).

- *Règles de dérivation générant des variables :*

```
<Intervention_acteur> ::= <Expression_besoin> <Suite_Intervention>
<Suite_Intervention> ::= [ <Séparateur> <Scénario_liste> ]
<Expression_besoin> ::= <Label_besoin> <Deux_points> <Liste_contraintes>
<Liste_contraintes> ::= { <Contrainte> }
<Contrainte> ::= <Attribut> <Comparateur> <Valeur>
<Attribut> ::= <Chaîne>
<Chaîne> ::= ( <Lettre> )+
```

- *Règles de dérivation générant des tokens :*

```
<Séparateur> :: = '&'
<Label_besoin> :: = 'Besoin'
<Deux_points> :: = ':'
<Comparateur> :: = '<' | '>' | '=' | '<>'| '<=' | '>='
<Valeur>:: = '0'..'9'
<Lettre> :: = 'a'..'z' | 'A'..'Z'
```

I.2. Classement des contraintes

La gestion d'informations (contraintes) structurées nécessite le recours à des formalismes spécifiques, en l'occurrence les structures linéaires (listes). Une liste de contraintes est une suite finie, éventuellement vide, et ordonnée de contraintes décideurs repérées selon leur rang.

I.2.1. Paramètres de classement des contraintes

Les contraintes sont représentées chacune sous forme d'un regroupement de cinq paramètres : L'identifiant de la contrainte, sa spécification textuelle[55], sa valeur de préférence, sa complexité et sa pertinence métier. Ces trois derniers paramètres représentent les paramètres utilisateur, système et métier, qui servent à déterminer l'ordre des contraintes dans le plan de satisfaction.

[55] L'identifiant et la spécification de la contrainte ne sont pas considérés ici pour des raisons de simplicité.

- ***Paramètre Utilisateur*** relatif au **N**iveau de **P**référence (NP) des contraintes : Ce paramètre, déduit de l'expertise décideur (classement suivant le critère préférence sur les contraintes), reflète l'importance de chaque contrainte pour le décideur et la nécessité de la satisfaire par le système. Il permet ainsi de classer les contraintes par niveau hiérarchique de préférence afin de privilégier la satisfaction des contraintes les plus préférées aux décideurs.

- ***Paramètre Système*** relatif au niveau de complexité[56] des contraintes : Ce paramètre, déduit de l'expertise conceptuelle (classement suivant le critère liens conceptuels), reflète la complexité temporelle des contraintes. Il permet ainsi de classer les contraintes par ordre croissant de leur complexité afin de traiter en premier les moins coûteuses en ressources ou d'offrir à l'utilisateur le moyen de les négocier en connaissance de cause.

- ***Paramètre Métier*** relatif à la pertinence métier des contraintes : Ce paramètre, déduit de l'expertise métier (critère nature des contraintes), exprime le poids absolu représentatif de chaque contrainte et son degré de représentativité par rapport à toute l'application. La détermination de ce paramètre passe par les étapes suivantes :

  - La répartition des contraintes suivant les familles : spatiales, temporelles et alphanumériques. Les contraintes dérivées seront représentées simultanément dans plusieurs familles suivant leur composition.
  - L'attribution d'un coefficient métier $representation_{FA}$ pour chaque famille F_A de contraintes conformément à son degré de représentativité par rapport à l'environnement métier de l'application.
  - L'attribution d'un poids relatif p_{ci} pour chaque contrainte c_i correspondant à son poids dans sa famille respective.

La pertinence métier $pertinence_{Ci}$ d'une contrainte c_i se calcule en fonction des deux coefficients p_{ci} et $representation_{Fci}$: $pertinence_{ci} = representation_{Fci} \times p_{ci}$ avec $representation_{Fci}$ le coefficient métier de la famille de la contrainte c_i. La pertinence métier d'une contrainte dérivée[57] est équivalente à une agrégation (e.g., somme) des pertinences métier calculées séparément par rapport à chaque famille d'appartenance.

[56] Ce paramètre de complexité sera traité plus en détails dans la suite de ce chapitre (partie II.1 : Analyse de la complexité).

[57] Dans le cas d'une contrainte dérivée appartenant à deux familles *F1* et *F2*, de coefficients métier respectifs $representation_{F1}$ et $representation_{F2}$, ayant un poids relatif p_1 et p_2 respectivement dans *F1* et *F2* et en utilisant la somme comme fonction d'agrégation : $pertinence_{Ci} = (p_1 \times representation_{F1}) + (p_2 \times representation_{F2})$.

I.2.2. Formalisme de représentation des contraintes

Pour représenter les besoins des décideurs et les différents paramètres (utilisateur, système et métier) définis ci-dessus, nous utilisons un formalisme de type abstrait de données TAD[58] [WIRTH 86]. Un TAD est défini à partir des domaines de base, à savoir : Entier, Réel, Chaîne de caractères, Booléen, Temps et de constructeurs d'agrégation [], d'ensemble {}, de liste <> et d'énumération de constantes de type (). Sa définition récursive est la suivante :

Soit a_i des noms d'attributs différentiables des noms de type.

Si $T_1 \ldots T_n$ sont des types alors $T = [a_1: T_1, \ldots, a_n: T_n]$ est un type ;

Si T_1 est un type alors $T= \{T_1\}$ est un type ;

Si T_1 est un type alors $T= \langle T_1 \rangle$ est un type ;

Si $T_1 \ldots T_n$ sont des constantes de type alors $T = (T_1, \ldots, T_n)$ est un type.

Nous considérons une structure `contraintesApplicatives` (figure 19), reflétant le plan de satisfaction des contraintes décideur, et qui représente une agrégation (induite du paramètre utilisateur) de trois Niveaux de Préférences (NP) relatifs respectivement aux contraintes dures, flexibles et secondaires :

```
Type contraintesApplicatives =
[
   Dures : NP,
   Flexibles : NP,
   Secondaires : NP
]
```

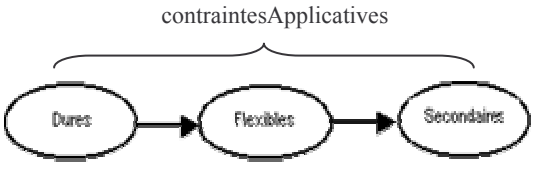


Figure 19 : Représentation de la structure `contraintesApplicatives`.
Les flèches indiquent le sens du passage du NP prioritaire au moins prioritaire.

Chaque `NP` (figure 20) représente une agrégation (induite du paramètre système) de trois niveaux de complexité, relatifs à la complexité d'évaluation en nombre d'opérations vis à vis

[58] Un type abstrait de données sert à manipuler des données de façon abstraite.

du nombre de données manipulées. Nous distinguons les niveaux de complexité constante, linéaire (complexité $1<\Phi\leq n$) et non linéaire/surélevée[59] ($\Phi>n$) :

```
Type NP =
[
   Constantes : niveauComplexité,
   Linéaires : niveauComplexité,
   Surélevées : niveauComplexité
]
```

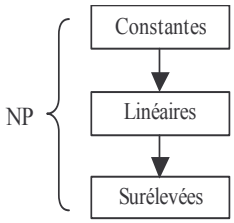


Figure 20 : Représentation de la structure `NP`. Les flèches indiquent le sens du passage du niveau de complexité faible au plus élevé.

La structure `contraintesApplicatives` représente donc une agrégation d'agrégation de trois niveaux de complexité (figure 21):

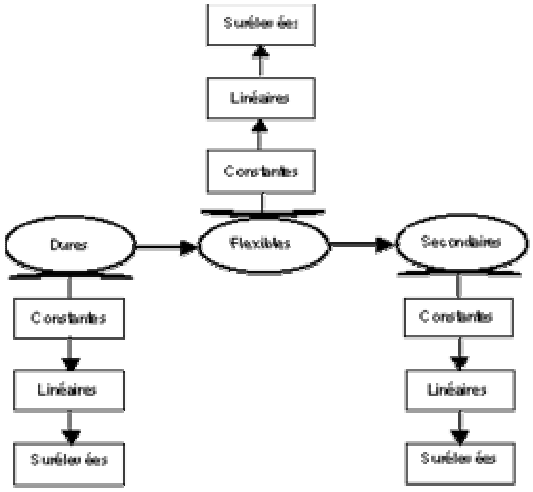


Figure 21 : Représentation de la structure `contraintesApplicatives` suivant les niveaux de préférences et de complexité.

Chaque niveau de complexité (`niveauComplexité`) est une collection non ordonnée (ensemble) de contraintes décideur. L'introduction du critère de pertinence métier (induit du paramètre métier) réordonne les contraintes à l'intérieur de chaque niveau de complexité de façon à avoir les contraintes les plus pertinentes en tête.

[59] Regroupant les autres types de complexité non linéaire, entre autres : nlogn, n^2, n^3, etc.

Chaque niveau de complexité représente ainsi une liste de contraintes ordonnées (par ordre décroissant) suivant leur pertinence métier (et par conséquent le `NP` représente une agrégation de liste de contraintes) :

`Type` *`niveauComplexité`* `=⟨ contrainteDécideur ⟩.`

Nous représentons formellement une `contrainteDécideur` (figure 22) par une agrégation de cinq composants, selon ce formalisme :

```
Type contrainteDécideur =
[
   identifiant : Chaîne de caractères
   spécification : Chaîne de caractères
   valeurPréférence : préférenceDécideur
   valeurComplexité : complexitéConceptuelle
   pertinence : Réel
]
```

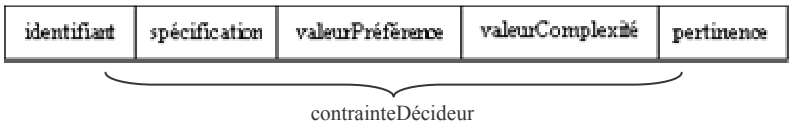


Figure 22 : Représentation d'une `contrainteDécideur`.

Les types `préférenceDécideur` et `complexitéConceptuelle` représentent des types énumérés (séquence de valeurs désignées par des constantes de type) définis comme suit :

`Type` *`préférenceDécideur`* `= (DURE, FLEXIBLE, SECONDAIRE)` avec `DURE`, `FLEXIBLE` et `SECONDAIRE` sont des constantes de type `préférenceDécideur`.

`Type` *`complexitéConceptuelle`* `= (CONSTANTE, LINEAIRE, SURELEVEE)` avec `CONSTANTE`, `LINEAIRE` et `SURELEVEE` sont des constantes de type `complexitéConceptuelle`.

Le plan de satisfaction ainsi obtenu, représente une agrégation (selon la valeur de préférence) d'agrégation (selon la valeur de complexité) de liste de contraintes ordonnées suivant la pertinence métier. Il peut être assimilé à une liste de contraintes ordonnées suivant la valeur de préférence, la valeur de complexité et la pertinence métier, et définir ainsi un ordre total :

`planSatisfaction` $= \langle c^{i}_{jk} \rangle$ avec c^{i}_{jk} : la contrainte appartenant au Niveau de Préférence *i*, de niveau de complexité *j* et de rang *k* (ordre décroissant suivant la pertinence métier) dans le niveau de complexité *j*.

Ce plan de satisfaction reflète les contraintes des décideurs, ordonnées suivant trois paramètres : métier, système et utilisateur. Il permet ainsi d'établir un ordre de satisfaction des contraintes et peut être représenté comme suit (figure 23) :

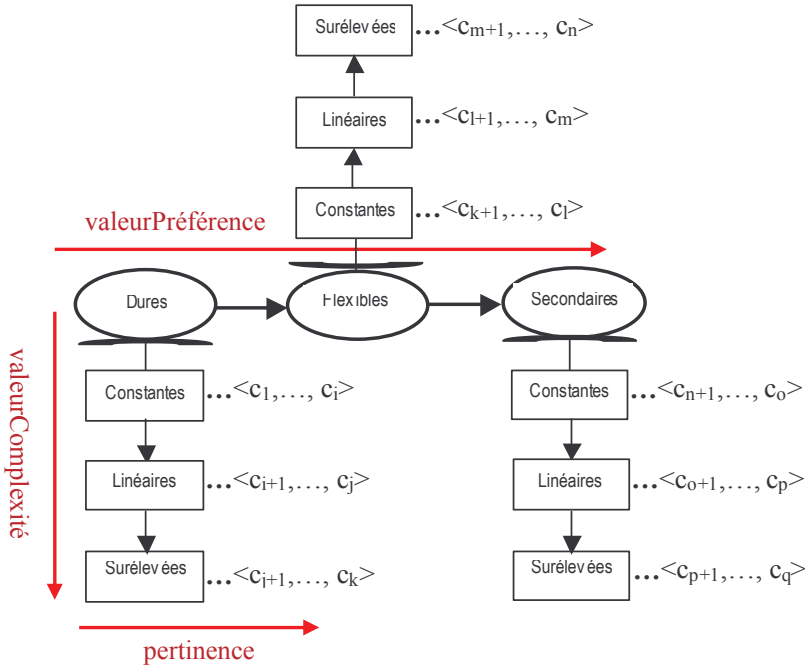


Figure 23 : Plan de satisfaction des contraintes.

I.3. Négociation des contraintes

Dans notre contexte de gestion globale du territoire agricole, la recherche d'un compromis prenant en compte les exigences multiples des décideurs et la marge de manœuvre du système impose une concertation et collaboration entre les différents acteurs impliqués. Il ressort la nécessité d'intégrer les décideurs dans la prise de décision et l'expression de leurs points de vue. Cette intégration s'exprime naturellement à travers une phase de négociation stipulant l'interaction analyste-décideur.

Nous définissons ainsi une méthodologie d'interaction entre le système et les décideurs basée sur un mécanisme de négociation des contraintes. Ce mécanisme favorise l'orientation des choix et interventions des décideurs vers la préservation d'un système de contraintes de complexité raisonnable accordant une certaine marge de manœuvre au système. La phase de négociation a donc pour objectif d'équilibrer le rapport Besoins / Ressources et de guider

l'utilisateur dans l'expression de ses choix et dans sa prise de décision. Nous espérons ainsi offrir aux décideurs la possibilité de contrôler et d'apporter des modifications sur la liste des contraintes (au moyen d'opérations) et de visualiser les conséquences de leurs choix.

Le principe de la phase de négociation est d'assister et aider le décideur à relaxer ses contraintes plutôt que de les relaxer à sa place. A cet effet, nous définissons des opérations support de la négociation afin de donner aux décideurs et au système le moyen de contrôler et de gérer la phase de négociation en agissant sur la liste des contraintes. Les principales opérations consistent à modifier les préférences des décideurs sur une liste de contraintes ou à supprimer des contraintes suivant un critère relatif à leur pertinence, leur valeur de préférence ou leur complexité.

I.3.1. Principe de négociation

La phase de négociation est basée sur le principe « négocier pour convaincre ». En effet, pour satisfaire l'ensemble des contraintes exigées par les décideurs, le système doit disposer des ressources suffisantes nécessaires à leur satisfaction. Ces ressources exprimées en terme de temps de réponse peuvent être déterminées à l'aide du paramètre de complexité qui révèle le coût nécessaire pour la satisfaction de chaque contrainte et reflète donc la complexité temporelle de l'application.

Le décideur et le système mènent une négociation ayant pour finalité la recherche d'un compromis équilibrant le rapport : temps de réponse nécessaire (coût total des contraintes) / temps de réponse disponible (ressources système). L'équilibre de ce rapport de coût peut se faire en modifiant le système de contraintes et/ou le temps de réponse autorisé pour fournir une solution.

I.3.2. Manipulation dynamique des contraintes

La phase de négociation est guidée par le système et gérée par le décideur. Ce dernier intervient au début du processus pour exprimer ses besoins sous forme de contraintes et ses préférences sur ces besoins. A partir de cette première série de contraintes introduites par les décideurs, le système évalue la complexité temporelle totale de l'ensemble des contraintes et visualise ce coût au décideur. Ce dernier, en fonction des coûts relatifs aux contraintes qu'il exige et du temps d'attente qu'il autorise au système avant de fournir une solution, tente d'équilibrer le rapport entre ces deux paramètres. Un ensemble d'opérations sur le système de

contraintes est ainsi défini afin de permettre aux décideurs de réduire le nombre de contraintes à satisfaire ou d'augmenter le temps maximum autorisé.

Outre les opérateurs de manipulation conventionnels des listes : créerListe (), listeVide (), rechercherElément (), supprimerElément (), etc., les opérateurs ensemblistes de base comme l'union ou l'intersection ainsi que les opérateurs logiques : $\wedge$, $\vee$, $\neg$, nous définissons des opérations primitives de manipulation du type *contrainteDécideur*, des opérations complexes manipulant le plan de satisfaction et les différentes structures définies dans la grammaire des contraintes et des opérations dérivées pouvant être exprimées en fonction d'autres opérations. Nous faisons ici abstraction de la phase de saisie et de visualisation des contraintes des décideurs en s'intéressant plus particulièrement au processus de manipulation et de mise à jour de ces contraintes.

Pour illustrer les opérations de manipulation des contraintes, nous nous basons sur les structures de référence : *contraintesApplicatives* reflétant le plan de satisfaction et *listeContraintes* représentant une liste d'éléments de type *contrainteDécideur* :

Type *listeContraintes* = ⟨ contrainteDécideur ⟩.

Pour l'illustration des opérations à partir d'exemples, nous nous plaçons dans le contexte (environnement) opérationnel suivant :

- lesContraintes1, lesContraintes2 : variables de type `contraintesApplicatives` ;
- contraintesSecondaires : variable de type `NP` ;
- contraintesSurélevées : variable de type `niveauComplexité` ;
- liste1, liste2 : variables de type `listeContraintes` ;

Nous utilisons dans ce qui suit les notations suivantes :

→ : Retour de l'opération ;

× : Séparateur des paramètres de l'opération.

Exemple : listeContraintes × Entier → listeContraintes signifie que l'opération admet deux paramètres de type listeContraintes et Entier et retourne un résultat de type listeContraintes.

Les principales opérations primitives sont les suivantes :

- ***Insérer :*** `listeContraintes × contrainteDécideur → listeContraintes.`

L'opération `Insérer` ajoute une contrainte décideur à la liste initiale suivant sa valeur de préférence, sa valeur de complexité et sa pertinence. Elle retourne une liste de longueur supérieure à la liste initiale.

- ***Supprimer :*** `listeContraintes × contrainteDécideur → listeContraintes.`

L'opération `Supprimer` retire une contrainte décideur de la liste initiale. Elle retourne une liste de longueur inférieure à la liste initiale. Cette opération n'est pas définie si la contrainte à supprimer n'appartient pas à la liste initiale des contraintes.

- ***Modifier :*** `contrainteDécideur × préférenceDécideur → contrainteDécideur.`

L'opération `Modifier` met à jour la valeur de préférence d'une contrainte. Cette opération retourne une contrainte de même valeur de complexité et pertinence métier.

- ***Sélectionner :*** `listeContraintes × conditionSélection → listeContraintes.`

L'opération `Sélectionner` permet d'extraire une liste de contraintes vérifiant une condition de sélection σ. La condition σ s'applique sur les paramètres[60] : identifiant, spécification et pertinence caractérisant les contraintes ou sur les variables liées par les contraintes. Une condition de sélection prend la forme d'un triplet : attribut, comparateur, valeur entourés par des guillemets (et suivis éventuellement par d'autres conditions séparées entre elles par des opérateurs logiques). Cette opération retourne une liste, éventuellement vide, de contraintes de longueur inférieure ou égale à la liste initiale.

Exemple : Sélectionner les contraintes de pertinence métier inférieure à 0,2 :
liste2 = `Sélectionner` (liste1, " pertinence <0,2 ").

Les principales opérations complexes à la disposition du décideur pour modifier le plan de satisfaction des contraintes sont : l'extraction de contraintes (`Extraire`), la suppression de contraintes (`Tailler`), l'insertion d'une liste de contraintes (`Insérer`), la mise à jour des

[60] Les paramètres valeurPréférence et valeurComplexité ne sont pas supportés par l'opération *Sélectionner*. La sélection suivant ces deux paramètres peut être exprimée via l'opération *Extraire*.

préférences des décideurs sur une liste de contraintes (`Modifier`), l'évaluation de la complexité temporelle des contraintes (`Evaluer`) et le respect d'un temps de simulation (`Respecter`).

- ***Extraire :*** `contraintesApplicatives × préférenceDécideur → NP.`

L'opération polymorphe `Extraire` permet d'extraire un niveau de préférence NP (correspondant à la préférence décideur spécifiée) de la structure contraintesApplicatives.

Exemple : Extraire les contraintes secondaires (figure 24) :
contraintesSecondaires = `Extraire` (lesContraintes1, SECONDAIRE).

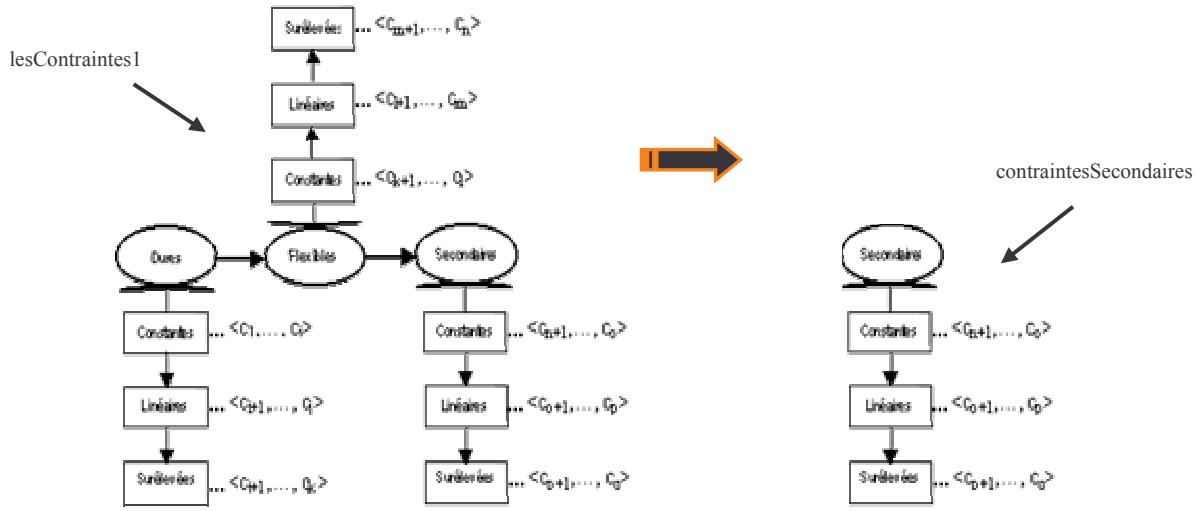


Figure 24 : Illustration de l'opération `Extraire`.

- ***Extraire :*** `NP × complexitéConceptuelle → niveauComplexité.`

L'opération polymorphe `Extraire` permet d'extraire un niveau de complexité (correspondant à la complexité conceptuelle spécifiée) relatif à un NP donné.

Exemple : Extraire les contraintes secondaires de complexité surélevée (figure 25) :
contraintesSurélevées = `Extraire` (`Extraire` (lesContraintes1, SECONDAIRE), SURELEVEE).

Figure 25 : Illustration de l'opération `Extraire`.

- ***Tailler*** : `contraintesApplicatives` × `préférenceDécideur` → `contraintesApplicatives`.

L'opération polymorphe `Tailler` revient à supprimer un niveau de préférence de la structure contraintesApplicatives, i.e., supprimer les contraintes de valeur de préférence égale à préférenceDécideur. Cette opération retourne une structure, éventuellement vide, de contraintes, incluse dans la structure initiale.

Exemple : Tailler les contraintes secondaires (figure 26) :

lesContraintes2 = `Tailler` (lesContraintes1, SECONDAIRE).

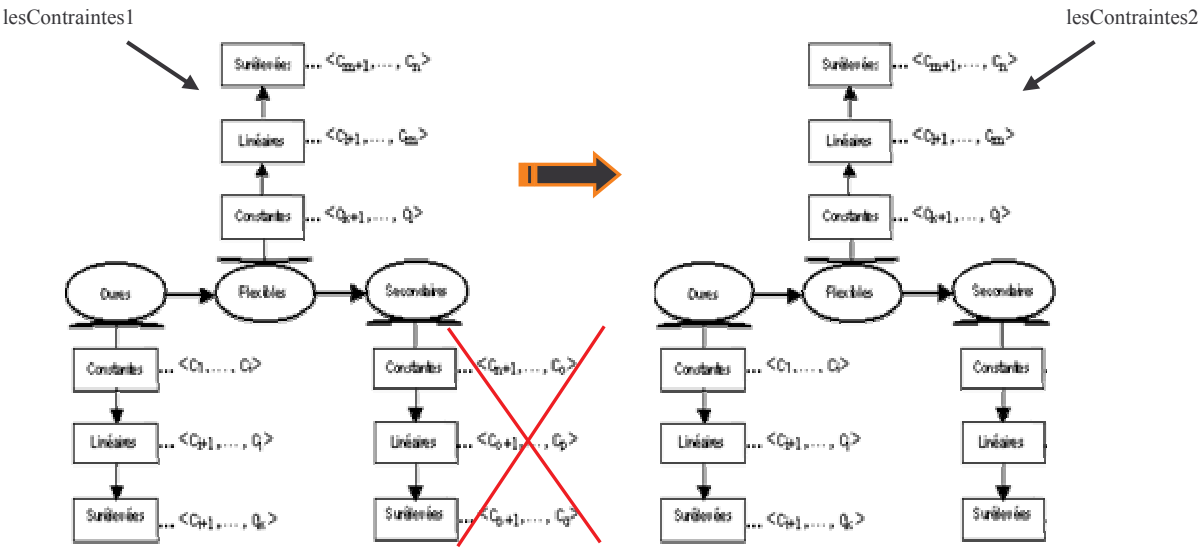


Figure 26 : Illustration de l'opération `Tailler`.

REMARQUE. — La suppression de plusieurs niveaux de préférence appartenant à la structure contraintesApplicatives est équivalente à une combinaison de plusieurs opérations `Tailler`.

Exemple : Tailler les contraintes secondaires ou flexibles (figure 27) :

lesContraintes2 = `Tailler` (`Tailler` (lesContraintes1, SECONDAIRE), FLEXIBLE).

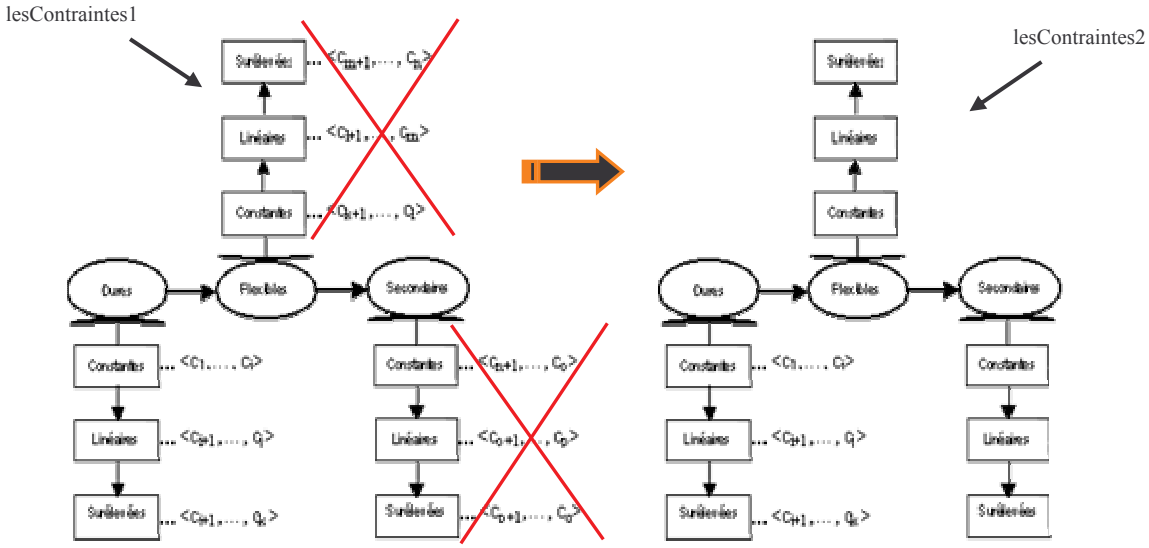


Figure 27 : Illustration de l'opération `Tailler`.

- ***Tailler :*** `NP × complexitéConceptuelle → NP.`

L'opération polymorphe `Tailler` revient à supprimer un niveau de complexité appartenant à un NP donné. Cette opération retourne un niveau de préférence NP, éventuellement vide, inclus dans le NP initial.

Exemple : Tailler les contraintes secondaires de complexité surélevée (figure 28) : contraintesSecondaires = `Tailler` (`Extraire` (lesContraintes1, SECONDAIRE), SURELEVEE).

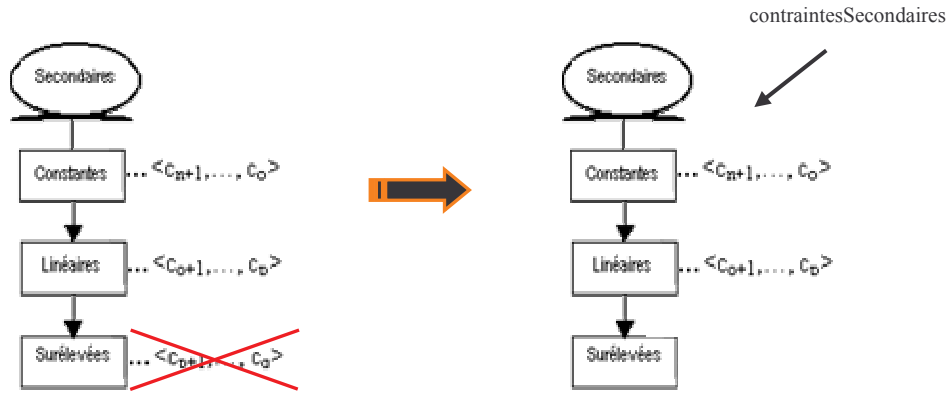


Figure 28 : Illustration de l'opération `Tailler`.

REMARQUE. — La suppression de plusieurs niveaux de complexité appartenant à un NP est équivalente à une combinaison de plusieurs opérations `Tailler`, appliquées au même NP.

Exemple : Tailler les contraintes secondaires de complexité surélevée ou linéaire (figure 29) :

contraintesSecondaires = `Tailler` (`Tailler` (`Extraire` (lesContraintes1, SECONDAIRE), SURELEVEE), LINEAIRE).

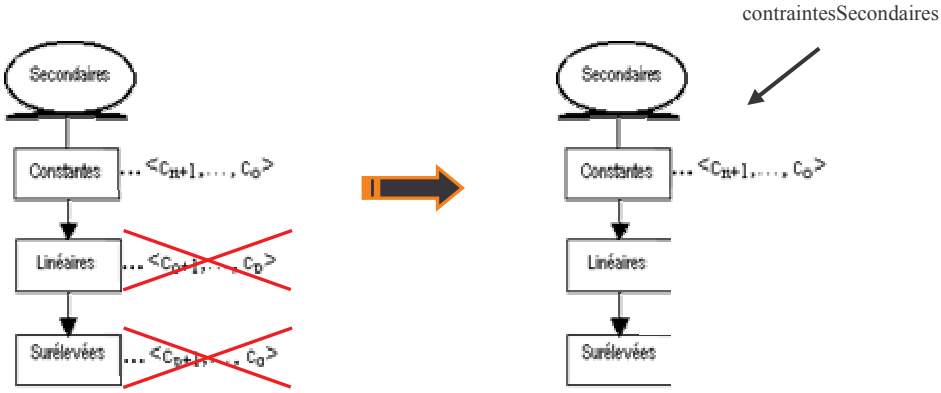


Figure 29 : Illustration de l'opération `Tailler`.

- ***`Tailler`*** **:** `contraintesApplicatives` × `complexitéConceptuelle` → `contraintesApplicatives`.

L'opération polymorphe `Tailler` revient à supprimer un niveau de complexité de la structure contraintesApplicatives, i.e., supprimer les contraintes de valeur de complexité égale à complexitéConceptuelle. Cette opération retourne une structure contraintesApplicatives, éventuellement vide, incluse dans la structure contraintesApplicatives initiale.

Exemple : Tailler toutes les contraintes de complexité surélevée (figure 30) :
lesContraintes2 = `Tailler` (lesContraintes1, SURELEVEE).

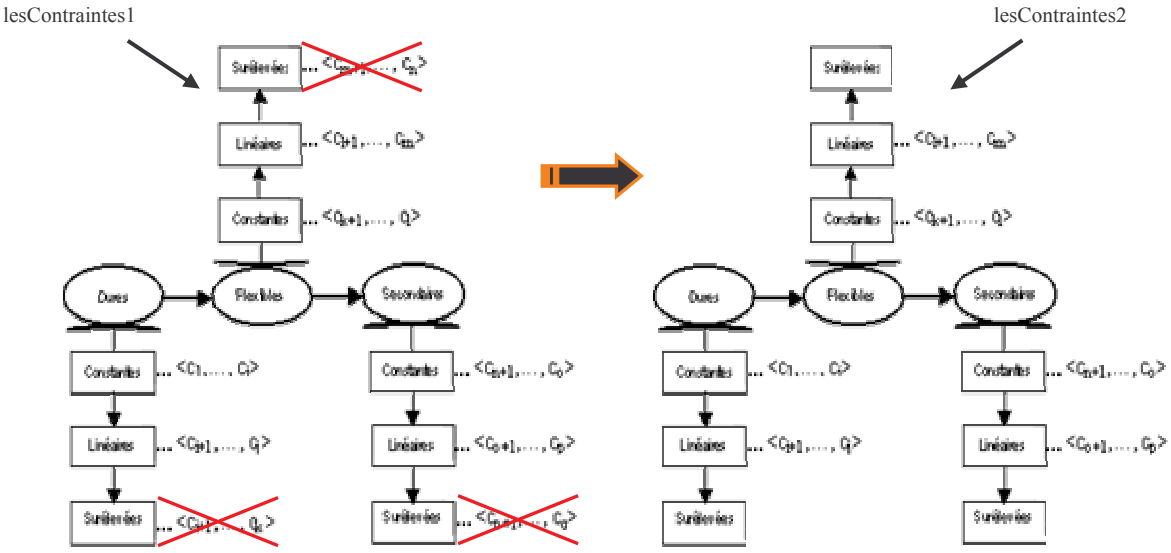


Figure 30 : Illustration de l'opération `Tailler`.

REMARQUE. — La suppression de plusieurs niveaux de complexité appartenant à une structure contraintesApplicatives est équivalente à une combinaison d'opérations élémentaires `Tailler`.

Exemple : Tailler toutes les contraintes de complexité surélevée ou linéaire (figure 31) :
lesContraintes2 = `Tailler` (`Tailler` (lesContraintes1, SURELEVEE), LINEAIRE).

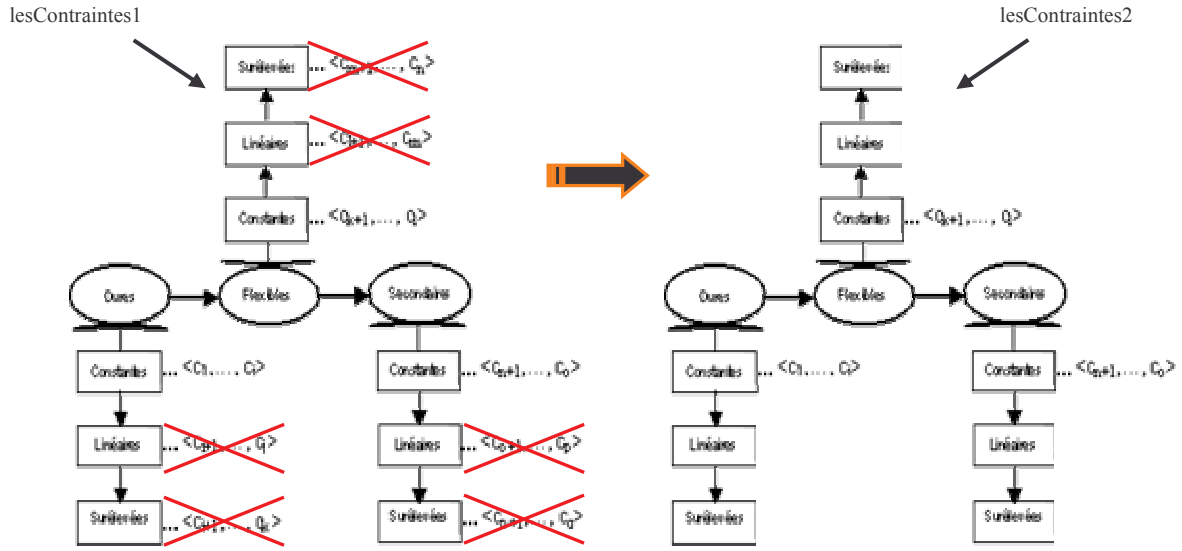


Figure 31 : Illustration de l'opération `Tailler`.

- ***Tailler*** : `contraintesApplicatives` × `listeContraintes` → `contraintesApplicatives`.

L'opération polymorphe `Tailler` revient à supprimer une liste de contraintes de la structure contraintesApplicatives. Cette opération retourne une structure contraintesApplicatives, éventuellement vide, incluse dans la structure de contraintes initiale. Elle n'est pas définie si les contraintes à tailler n'appartiennent pas à la structure contraintesApplicatives.

Exemple : Tailler dans la structure lesContraintes1, les contraintes de pertinence < 0,2 (figure 32) :
lesContraintes2 = `Tailler` (lesContraintes1, `Sélectionner` (lesContraintes1, " pertinence <0,2 ").

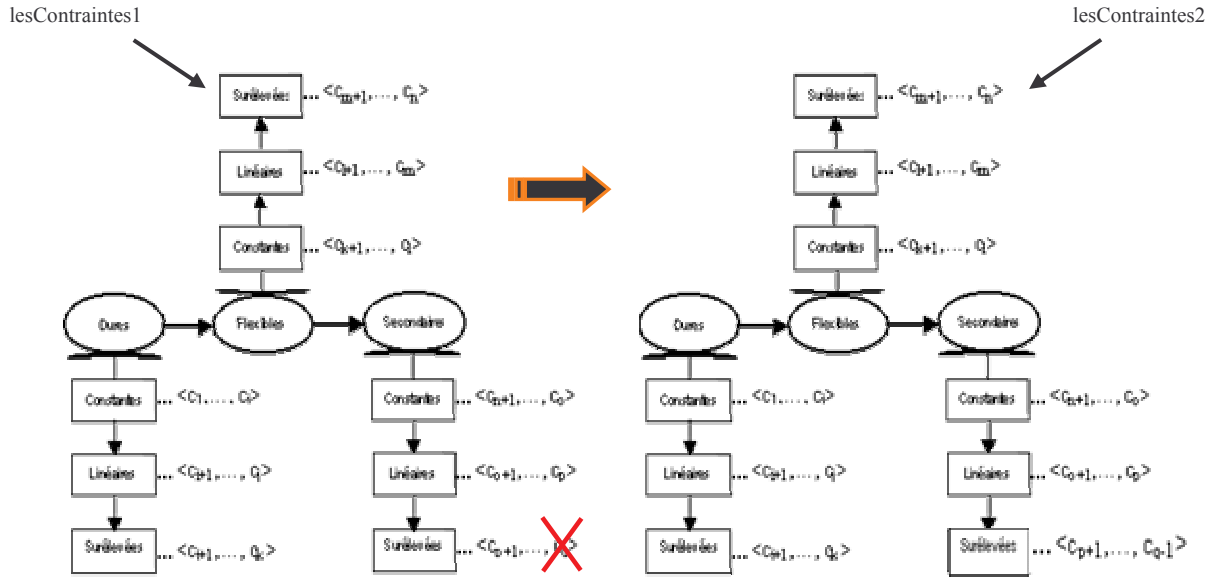


Figure 32 : Illustration de l'opération `Tailler`.

- ***Insérer :*** `contraintesApplicatives × listeContraintes → contraintesApplicatives.`

L'opération `Insérer` ajoute une liste de contraintes décideur à la structure contraintesApplicatives. Les contraintes ajoutées seront classées suivant leur valeur de préférence, leur valeur de complexité et leur pertinence métier. Cette opération retourne une structure contraintesApplicatives de cardinalité (de contraintes) supérieure à la structure initiale.

- ***Modifier :*** `listeContraintes × préférenceDécideur → listeContraintes.`

L'opération `Modifier` met à jour la valeur de préférence d'une liste de contraintes tout en gardant la même valeur de complexité et pertinence métier. Elle retourne une liste de contraintes de même longueur que la liste initiale.

- ***Evaluer :*** `contraintesApplicatives → Temps.`

L'opération `Evaluer` mesure le temps nécessaire pour satisfaire toutes les contraintes de la structure contraintesApplicatives. Cette opération retourne la complexité temporelle globale des contraintes de la structure.

- ***Respecter :*** `contraintesApplicatives × Temps → Booléen.`

L'opération `Respecter` revient à ne pas dépasser un temps maximum autorisé pour fournir une solution satisfaisant les contraintes de la structure contraintesApplicatives. Cette opération retourne un booléen relatif à l'état de satisfaction du temps exigé.

Nous définissons l'opération dérivée `Replacer`, permettant la réorganisation des contraintes suivant une nouvelle valeur de préférence :

- ***Replacer :*** `contraintesApplicatives × préférenceDécideur × complexitéConceptuelle × préférenceDécideur → contraintesApplicatives.`

L'opération `Replacer` revient à modifier, dans une structure contraintesApplicatives, le positionnement des contraintes vérifiant une préférenceDécideur et une complexitéConceptuelle données. Ceci nous amène à sélectionner, dans la structure contraintesApplicatives initiale, les contraintes vérifiant la préférenceDécideur et la complexitéConceptuelle spécifiées (opération `Extraire`), à mettre à jour leurs préférences (opération `Modifier`), les insérer (dans un autre Niveau de Préférence) suivant leur nouvelle valeur de préférence tout en gardant les mêmes valeurs de complexité et pertinence métier (opération `Insérer`) et enfin à supprimer les contraintes sélectionnées pour modification, de la structure contraintesApplicatives (opération `Tailler`). L'opération `Replacer` retourne une structure contraintesApplicatives de même cardinalité (nombre de contraintes) que la structure initiale.

Exemple : Replacer au Niveau de Préférence Secondaires, les contraintes de complexité surélevée et de valeur de préférence dure (figure 33) :
lesContraintes2 = `Replacer` (lesContraintes1, DURE, SURELEVEE, SECONDAIRE) ⇔

L'expression de la requête à l'aide des opérations `Tailler`, `Insérer`, `Modifier` et `Extraire` prend la forme suivante :
lesContraintes2 = `Tailler` (`Insérer` (lesContraintes1, `Modifier` (`Extraire` (`Extraire` (lesContraintes1, DURE), SURELEVEE), SECONDAIRE)), `Extraire` (`Extraire` (lesContraintes1, DURE), SURELEVEE)).

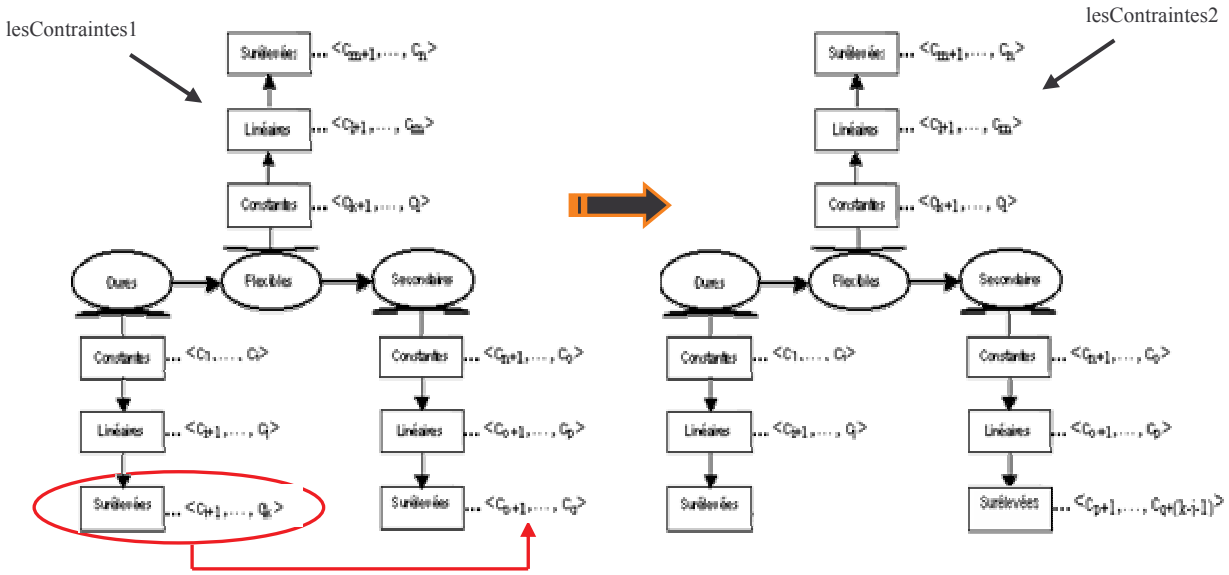


Figure 33 : Illustration de l'opération `Replacer`.

REMARQUE. — Le replacement d'un niveau de complexité appartenant à plusieurs niveaux de préférence est équivalent à une combinaison d'opérations `Replacer`.

Exemple : L'opération replacer au Niveau de Préférence Secondaires, les contraintes de complexité surélevée et de valeur de préférence dure ou flexible, se fait en deux temps (figure 34) :

lesContraintes2 = `Replacer` (`Replacer` (lesContraintes1, DURE, SURELEVEE, SECONDAIRE), FLEXIBLE, SURELEVEE, SECONDAIRE).

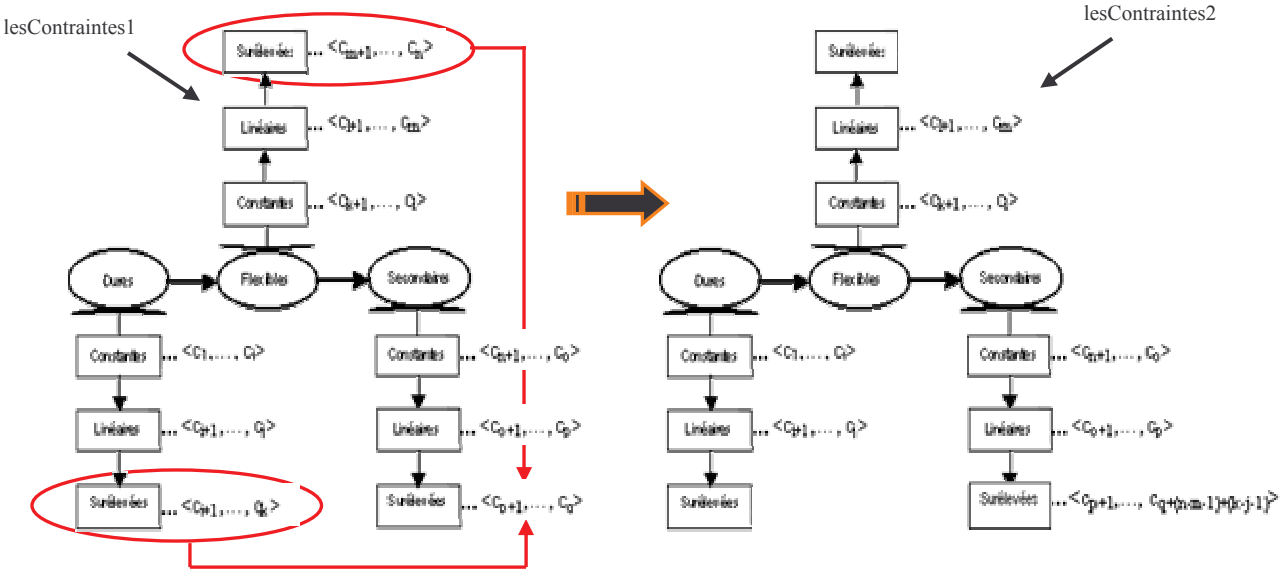


Figure 34 : Illustration de l'opération `Replacer`.

La phase de négociation permet d'établir, à partir du plan de satisfaction, le graphe opérationnel des contraintes effectives à prendre en compte par le système de simulation. Les opérations appliquées par le décideur sont contrôlées par le système afin de visualiser les conséquences des modifications (en terme de coût) apportées sur la liste des contraintes. En effet, le système joue la carte de la complexité des contraintes à satisfaire pour négocier avec le décideur et le convaincre de modifier ses choix et de proposer des désistements. Les choix et opérations effectués par les décideurs déterminent la marge de manœuvre accordée au concepteur et influent sur la qualité de la solution pouvant être proposée par le système de simulation.

I.4. Synthèse

Nous avons proposé dans cette partie une stratégie de négociation des contraintes, exprimant l'interaction entre le système et le décideur. Les figures 20 et 21 donnent une vue générale sur les différentes phases du processus d'acquisition et de modélisation dynamique des contraintes tel qu'il est présenté dans ce travail.

La mise en place d'un cadre générique pour traiter les problèmes d'optimisation sous contraintes nécessite une modélisation fidèle et adéquate des différents aspects du problème, notamment ceux liés à la complexité des contraintes exigées et au mécanisme adéquat pour assurer leur satisfaction tout en restant dans le cadre des objectifs visés par l'application. Ce point constitue le sujet de la partie suivante sur la modélisation de l'optimisation sous contraintes.

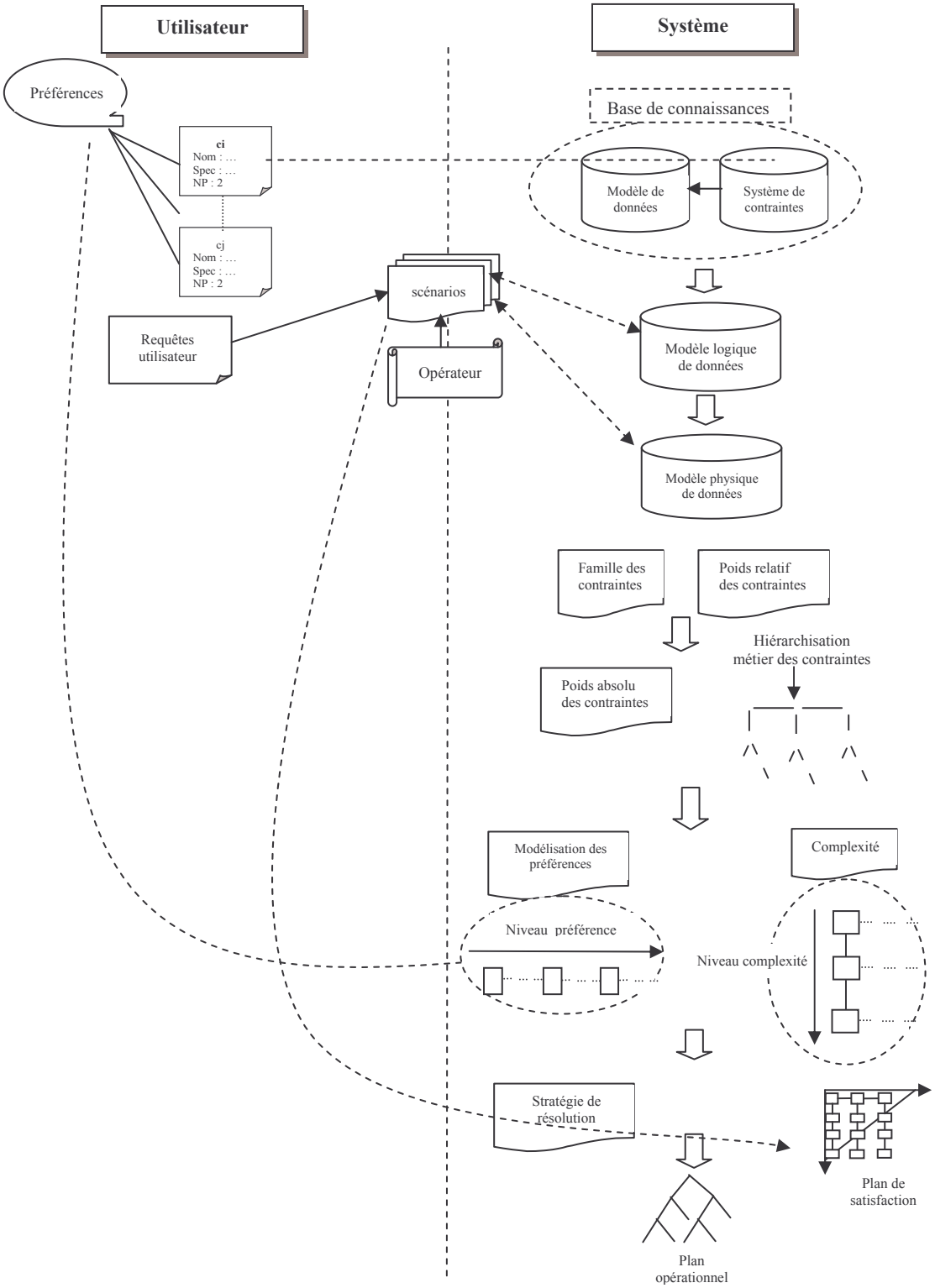


Figure 35 : Différentes phases de modélisation du système de contraintes.

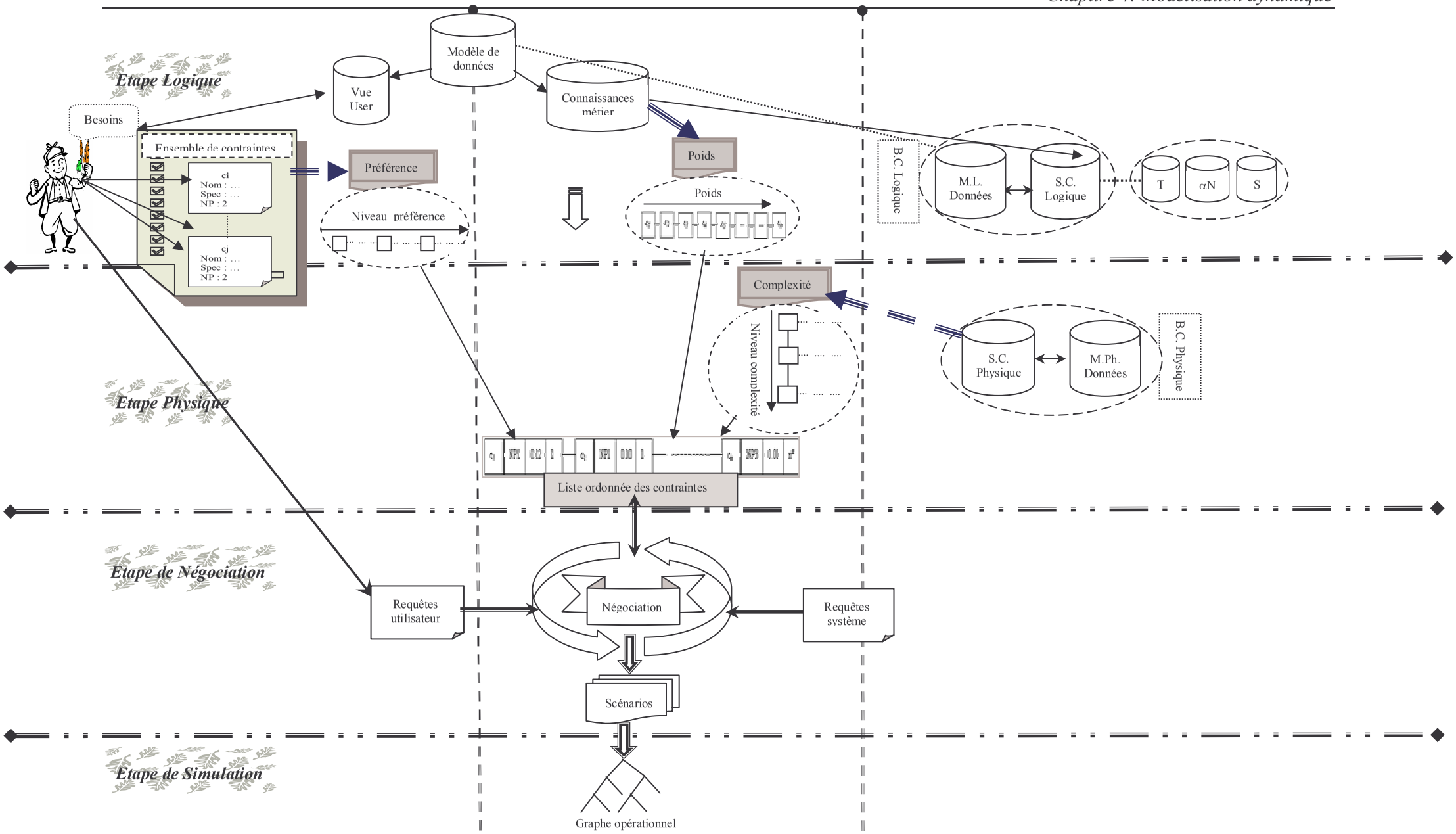


Figure 36 : Vue d'ensemble du processus de modélisation de l'interaction système/décideur.

II. Modélisation de l'optimisation sous contraintes

Nous étudions dans cette partie la complexité d'un système d'optimisation sous contraintes en nous basant sur l'analyse de la complexité des contraintes suivant le paramètre arité conceptuelle (défini dans la partie modélisation des contraintes du chapitre 3). La complexité de certaines contraintes peut s'avérer peu compatible avec la stratégie d'un algorithme d'optimisation. Afin de mieux gérer cette complexité, nous procédons à une classification des variables du système et à la formalisation de la structure des contraintes impliquant ces variables. La formalisation des contraintes permet de dégager les différentes classes de variables qui la composent et dont les valeurs prises déterminent la satisfaction de ces contraintes au cours du processus d'optimisation.

II.1. Analyse de la complexité

Dans les problèmes d'aménagement du territoire, les décideurs expriment naturellement beaucoup de contraintes dont la satisfaction nécessite souvent des traitements complexes, ce qui fait que les problèmes sont souvent sur-contraints. La détermination de la satisfiabilité du système : sous ou sur-contraint et de la marge de manœuvre du concepteur pour proposer des solutions satisfaisantes en un temps de calcul raisonnable nécessite le recours à une analyse de la complexité du problème traité. Cette analyse intègre nécessairement l'évaluation de la complexité des contraintes prises en compte et la détermination des traitements nécessaires pour leur satisfaction dans le cadre d'un processus d'optimisation.

Généralement, deux types de complexité peuvent être distingués : la complexité théorique et la complexité pratique. La complexité théorique d'un algorithme consiste à évaluer mathématiquement les ressources nécessaires pour son fonctionnement. La complexité pratique consiste à mesurer empiriquement le coût d'exécution de l'algorithme sur des données concrètes. Nous nous intéressons ici à la complexité théorique. La mesure de la complexité nécessite souvent une analyse mathématique détaillée selon la nature et l'ordonnancement des données manipulées afin de déterminer la complexité spatiale et temporelle. La complexité spatiale est l'évaluation de la place mémoire réservée à un traitement donné. La complexité temporelle quant-à-elle, est une évaluation du temps nécessaire à la recherche d'une solution. Elle s'exprime en fonction du nombre d'opérations et du temps mis pour chacune. Dans notre cas, seul l'ordre de grandeur du nombre

d'opérations est pris en compte. Les opérations qui, relativement aux autres, consomment peu de temps, telles que les tests (comparaisons), les affectations et les incrémentations ne sont pas comptabilisées.

Nous étudions dans cette partie la complexité d'un système d'optimisation sous contraintes en passant par la distinction de la complexité des traitements réels au niveau système de celle vue par l'utilisateur ainsi que l'analyse de la complexité des contraintes suivant le paramètre arité conceptuelle.

II.1.1. Complexité système et utilisateur dans les applications réelles

Dans les applications réelles, les traitements des utilisateurs sont en général simples et ne reflètent pas la complexité effective des contraintes au niveau conceptuel. Les opérations complexes[61] de recherche d'informations pertinentes, telles que les jointures, peuvent être faites à l'avance à l'image des schémas externes (cliché ou clustering) dans un Système de Gestion de Bases de Données Relationnel[62]. Un schéma externe d'un utilisateur ou d'un groupe d'utilisateurs est défini comme étant un sous-ensemble avec éventuellement une réorganisation, du schéma conceptuel modélisant la partie des données présentant un intérêt pour cet utilisateur. La structure du schéma externe peut être différente de celle du schéma conceptuel, ainsi que le modèle de données employé. L'intérêt des schémas externes réside notamment dans la protection des données contre les accès non autorisés et l'indépendance des programmes des utilisateurs vis-à-vis des modifications apportées au schéma conceptuel, telles que l'ajout d'attributs, etc.

II.1.2. Complexité des contraintes

La complexité d'une contrainte peut être exprimée en fonction de son arité conceptuelle (mono ou multi-instances), du nombre $\boldsymbol{e}$ d'entités concernées par la contrainte, du nombre $\boldsymbol{n}$ d'instances de chaque entité et du nombre $\boldsymbol{i}$ d'instances impliquées dans la contrainte à un instant donné ($\boldsymbol{i} \leq \boldsymbol{n}$).

[61] Ces opérations peuvent donner en résultat des relations qui ne sont pas normalisées puisque de toutes façons les relations externes sont fictives et ne sont pas concrétisées (ce ne sont que des fenêtres sur la Base de Données).

[62] En relationnel, un schéma externe est constitué d'un ensemble de relations externes qui sont déduites des relations du schéma conceptuel (et éventuellement d'autres vues) par des règles de correspondance.

Le nombre effectif ***i*** d'instances impliquées dans une contrainte à un instant donné est indépendant de son arité conceptuelle (indépendante du temps). En effet, une contrainte peut être multi-instances et impliquer une seule instance à un instant t ou inversement.

Exemple 1 : La somme des surfaces des parcelles contenant le lin doit être supérieure à 3 hectares. L'arité conceptuelle de la contrainte est multi-instances, i.e., la vérification de la contrainte nécessite la vérification simultanée (à la fois) de toutes les parcelles contenant le lin. Cependant, il peut y avoir une seule parcelle contenant le lin à un instant t (le nombre effectif d'instances peut être égal à 1).

Exemple 2 : Les parcelles situées à l'exutoire doivent contenir de l'herbe. Cette contrainte est mono-instance (nous pouvons vérifier si chaque parcelle (seule) située à l'exutoire contient de l'herbe, indépendamment des autres instances) bien qu'elle concerne toutes les parcelles situées à l'exutoire (le nombre effectif d'instances peut être supérieur à 1).

Cette complexité dépend ainsi de la clé ou de la pseudo-clé[63], des entités impliquées dans chaque contrainte. Pour chaque entité impliquée dans la contrainte, si la clé est donnée, alors l'accès est une opération d'indexation relative à une complexité constante, sinon l'accès est une opération de sélection de complexité linéaire. Nous proposons dans cette partie une analyse de la complexité des contraintes en se basant sur le paramètre arité conceptuelle qui établit une distinction des contraintes suivant le nombre d'entités et d'instances concernées. Nous ne prétendons pas offrir une étude fine et complète de la complexité dont l'analyse détaillée nécessite certainement des études plus élaborées et des recherches plus approfondies. Nous cherchons simplement à montrer la multitude des représentations possibles des contraintes pouvant être exigées par les décideurs et la diversité conséquente de leur mesure de complexité qui reste un aspect non négligeable dans la détermination et la limitation de la marge de manœuvre du concepteur pour l'élaboration des solutions satisfaisantes. L'aspect complexité peut ainsi constituer un critère efficace pour guider la phase de négociation des contraintes entre le système et les décideurs et sensibiliser ces derniers envers les problèmes rencontrés par le concepteur dans la modélisation et la gestion de la complexité des contraintes qu'ils exigent.

[63] Attribut disposant d'un index.

II.1.2.1. Contraintes intra-entité (e=1) :

L'évaluation de la satisfaction de ces contraintes se fait au sein d'une seule entité. Une contrainte intra-entité peut être mono ou multi-instances.

a. Contraintes mono-instance :

La satisfaction de la contrainte peut être vérifiée pour chaque instance indépendamment des autres.

- Si la contrainte induit la clé de l'entité (la clé est précisée) alors c'est une opération d'indexation par la clé. La complexité est constante : ***O(1)***. Exemple : Le sens de travail de la parcelle 5 doit être perpendiculaire à sa pente.

- Si la contrainte n'induit pas la clé (induit un attribut non-clé) alors c'est une opération de sélection[64]. La complexité est linéaire : ***O(n)***. Exemple : Ne pas dégrader la sensibilité des parcelles de profil financier important.

b. Contraintes multi-instances :

La vérification de la contrainte se fait en évaluant plus qu'une instance à la fois.

- Si la contrainte induit l'attribut clé pour toutes les ***i*** instances qu'elle implique alors son évaluation nécessite ***i*** (***i*** <<< ***n***) opérations d'indexation de complexité constante : ***O(i)*** ≈ ***O(1)***. Exemple : les parcelles 1 et 2 doivent contenir la même occupation.

- Si la contrainte n'induit pas la clé de l'entité alors la vérification de sa satisfaction nécessite une[65] opération de sélection. La complexité est linéaire : ***O(n)***. Exemple : Les cultures de type SCOP doivent avoir la même fréquence de retour.

REMARQUE. — D'autres contraintes multi-instances peuvent nécessiter des opérations plus complexes et atteindre ainsi une complexité quadratique (***n²***). Exemple : les parcelles très sensibles ayant un profil financier faible doivent avoir la même occupation que celles de pente > 10 ou ayant un type de sol dégradé.

[64] Accès sur des attributs non clés : parcours des n instances pour vérifier la condition cherchée.
[65] Nous nous limitons ici aux contraintes simples nécessitant une opération de sélection et nous supposons que les contraintes plus complexes peuvent être exprimées en fonction de ce type de contraintes.

II.1.2.2. Contraintes inter-entités (e>1) :

L'évaluation de la satisfaction de ces contraintes implique des données appartenant à des entités différentes. Nous supposons dans ce qui suit que les entités ont le même nombre ***n*** d'instances. Nous distinguons les contraintes mono-instance et multi-instances.

a. Contraintes mono-instance :

- Si la contrainte induit les clés des entités impliquées dans la contrainte alors son évaluation nécessite ***e*** opérations d'indexation (accès par les clés). La complexité est constante[66] : $O(e) \approx O(1)$. Exemple : Respecter la fréquence de retour du blé sur la parcelle 5.

- Si la contrainte induit ***s*** clés, avec $s < e$ (les clés de ***e-s*** entités ne sont pas données) alors son évaluation nécessite ***s*** opérations d'indexation + ***e-s*** opérations de jointure. La complexité est fonction du nombre ***e-s*** de clés non données : $O(s+(e-s) \times n^2) \approx O((e-s) \times n^2)$ (complexité quadratique au meilleur des cas : ***e-s*** =1). Exemple : L'herbe et le maïs ne doivent pas être placés loin du corps de ferme de l'exploitation.

- Si la contrainte n'induit pas les clés des entités[67] (***s*** = 0), deux cas se présentent. Si la contrainte ne comporte aucune condition de sélectivité, alors son évaluation nécessite ***e-1*** opérations de jointure. La complexité peut être exprimée par $O((e-1) \times n^2$ (complexité n^2 au meilleur des cas : e=2). Exemple : toutes les parcelles doivent contenir des cultures anti-ruisselantes. Si la contrainte comporte une condition de sélectivité alors son évaluation nécessite ***1*** opération de sélection et ***e-1*** opérations de jointure. La complexité est fonction du nombre ***e*** d'entités impliquées : $O(n+(e-1) \times n^2) \approx O((e-1) \times n^2)$ (complexité $n+n^2$ au meilleur des cas : e=2). Exemple : les parcelles sensibles appartenant à l'exploitation X, doivent contenir des cultures anti-ruisselantes.

b. Contraintes multi-instances :

Les contraintes inter-entités multi-instances sont généralement plus complexes que les autres types de contraintes et peuvent nécessiter des opérations de complexité > n^2. Exemple : Le chemin que doivent suivre les vaches appartenant à monsieur X ne doit pas croiser celui des vaches de ses voisins.

[66] Nous supposons que e <<<n et donc e ≈ 1.
[67] Ce cas peut être considéré comme un cas particulier du précédent avec s=0.

II.1.3. Complexité d'un système d'optimisation sous contraintes

Les problèmes d'optimisation peuvent atteindre rapidement une grande complexité et nécessiter des temps de calcul considérables à cause de l'explosion combinatoire du nombre des solutions à envisager [T'KINDT 02]. La complexité totale d'un problème d'optimisation sous contraintes peut être calculée en fonction du nombre t d'itérations[68] de l'algorithme et de la complexité C_t de chacune. La complexité d'une itération peut être exprimée en fonction de la complexité C_s de satisfaction, de la complexité C_c d'évaluation des contraintes et de la complexité C_o d'optimisation.

- La complexité C_s de satisfaction de contraintes représente le nombre d'opérations nécessaires pour l'affectation (ou la ré-affectation) des valeurs à des variables afin de satisfaire les contraintes. Cette complexité peut être déterminée en fonction du nombre c de contraintes à satisfaire au cours de l'itération, du nombre total v de variables liées par ces contraintes et de la taille d de leurs domaines (nous supposons que les domaines des variables sont tous égaux). Le nombre de tuples possibles (combinaisons possibles de valeurs des variables) = d^v. Chaque tuple qui satisfait les c contraintes étant une solution au problème, la complexité temporelle est de l'ordre de $O(d^v c)$.

- La complexité conceptuelle C_c d'évaluation reflète le temps nécessaire pour évaluer l'ensemble des contraintes prises en compte dans le processus d'optimisation. Cette complexité peut être déterminée via le paramètre arité conceptuelle des contraintes.

- La complexité C_o d'optimisation s'exprime en fonction du nombre d'opérations élémentaires nb_o nécessaires[69] pour atteindre l'objectif et le temps d'exécution t_o, dépendant du processeur, mis par chacune[70]. Il faut signaler que le nombre d'opérations élémentaires peut varier d'une itération à une autre (certaines données peuvent changer au cours de la résolution), notamment dans les environnements de résolution évolutifs ou instables ou encore pour les systèmes non déterministes (basés sur l'aléatoire).

La complexité totale d'optimisation sous contraintes peut être exprimée par :

$O(t \times C_t) = O(t \times (C_s + C_c + C_o))$.

Nous avons traité tout au long de ce travail trois aspects complémentaires : un aspect lié à la définition et la formulation des besoins des décideurs (partie modélisation de l'optimisation

[68] Le nombre d'itérations peut être fixé à l'avance (condition d'arrêt de l'algorithme) ou déduit à la fin de l'exécution (compteur d'itérations).
[69] Dépend de l'algorithme de résolution utilisé.
[70] Dépend des performances de la machine utilisée : 1/ fréquence Mhz.

du chapitre 3), un aspect orienté satisfaction lié à l'analyse et la classification des contraintes à satisfaire (partie modélisation des contraintes du chapitre 3) et un aspect orienté aide à la décision intégrant les décideurs dans le choix et la négociation de leurs contraintes (partie modélisation de l'interaction du chapitre 4). Notre travail entre dans le cadre des problèmes mixtes d'optimisation et de satisfaction, comportant à la fois des contraintes fortes à satisfaire et des critères multiples à améliorer. Il est généralement difficile d'assurer simultanément ces tâches de satisfaction et d'optimisation et la résolution bascule souvent vers l'une ou l'autre des deux tâches.

Les décideurs exigent des contraintes diverses dont la complexité système peut s'avérer peu compatible avec la stratégie d'un algorithme d'optimisation. L'analyse de la complexité des contraintes que nous venons d'évoquer ci-dessus, conforte cette analyse sans apporter une solution à ce problème d'incompatibilité. L'enjeu demeure de pouvoir agir face à une telle situation afin de mieux gérer la complexité des contraintes exigées par les décideurs. Compte tenu de ces exigences, notre réflexion nous a conduit à définir un cadre générique de résolution des problèmes d'optimisation sous contraintes. Nous cherchons plus précisément à orienter la résolution dans le cadre d'un processus mixte assurant à la fois la tâche d'optimisation et la tâche de satisfaction des contraintes tout en évitant (dans la mesure du possible) de favoriser l'une vis-à-vis de l'autre.

II.2. Formalisation et modélisation des contraintes pour l'optimisation

Les applications d'aménagement du territoire comportent souvent des tâches d'optimisation et de satisfaction des contraintes. Une stratégie d'optimisation est basée généralement sur l'exploration de l'espace de recherche, par la modification des valeurs de certaines variables système, dans le but de trouver la meilleure solution améliorant une fonction objectif. Les contraintes représentent des restrictions sur les valeurs qui peuvent être prises simultanément par les variables liées (par ces mêmes contraintes). Si nous voulons optimiser tout en respectant les contraintes, nous risquons (et c'est souvent le cas) d'être confrontés à des contraintes complexes rendant compliquée la tâche d'optimisation et limitant considérablement le processus d'exploration des solutions. Comment donc gérer ce paradoxe et pouvoir optimiser tout en satisfaisant les contraintes au cours d'un processus d'optimisation?

Après l'analyse du problème et une présentation de la particularité de notre étude par rapport aux approches classiques d'optimisation, nous étudions la stratégie d'optimisation adoptée dans ce travail. Nous proposons ensuite une classification des variables du problème suivant leur intervention dans le processus d'optimisation et leur dépendance vis-à-vis des contraintes. Les variables déterminent la satisfaction des contraintes dans lesquelles elles sont impliquées. La re-formulation de la structure des contraintes du point de vue de l'optimisation, permet de distinguer leurs propriétés et de dégager celles susceptibles d'évoluer à la suite de la ré-affectation des variables d'optimisation. Ainsi, il devient possible d'analyser l'implication des variables dans les contraintes et de suivre l'impact de la modification de leurs valeurs sur l'évolution de la satisfaction des contraintes au cours de l'optimisation.

II.2.1. Analyse du problème

Dans notre contexte d'optimisation sous contraintes naturellement multicritère et sur-contraint, nous disposons également d'un grand nombre de variables. Un tel problème ne peut être traité avec une méthode exacte et l'application d'une méthode approchée d'optimisation paraît la seule voie envisageable. Cependant, un processus d'optimisation fondé sur une méthode approchée nécessite naturellement une certaine flexibilité (souplesse) pour pouvoir explorer l'espace de recherche et il est donc généralement peu adapté aux problèmes comportant des contraintes fortes. Comment donc passer par un algorithme d'optimisation et veiller en parallèle à assurer la tâche de satisfaction des contraintes, indispensable pour proposer des solutions satisfaisantes aux décideurs ?

Les réflexions menées dans la littérature autour des problèmes d'optimisation sous contraintes ont contribué à la conception de certaines méthodes de résolution, telle que branch and bound [LAWLER 96]. Ces approches se basent essentiellement sur l'utilisation d'heuristiques au cours de la résolution pour orienter la recherche de solutions. La qualité des solutions apportées reste variable en fonction des heuristiques utilisées et dépend largement des informations spécifiques au problème traité. Les techniques de pré-traitement pour leur part, ont été développées pour réduire l'espace de recherche et accroître l'efficacité de résolution des problèmes faisant intervenir des contraintes [KUMAR 92]. Ces techniques, utilisées en combinaison avec une autre méthode de recherche, agissent au niveau des variables pour éliminer des inconsistances partielles sans toutefois se préoccuper du rôle de ces variables vis-à-vis de l'optimisation.

Bien qu'elles permettent d'éviter une énumération coûteuse de l'ensemble des solutions potentielles, la plupart des approches développées pour résoudre les problèmes d'optimisation sous contraintes restent limitées à l'analyse du problème au niveau du processus de recherche de solutions et ne descendent pas aux causes principales des insatisfactions. Le rôle joué par les variables dans les processus d'optimisation et de satisfaction est rarement mis en évidence pour assurer ces deux tâches simultanément et la résolution favorise plutôt l'une ou l'autre de ces tâches.

Une approche d'optimisation repose généralement sur l'affectation de valeurs à des variables (cf., les méthodes approchées) dans le but de trouver la bonne combinaison de valeurs améliorant le critère recherché. Le processus de recherche de solutions met en œuvre une stratégie d'exploration des données, qui dépend de l'algorithme appliqué, guidée par l'évolution du critère à améliorer. Une approche de satisfaction quant à elle, consiste à affecter des valeurs à des variables dans le but de satisfaire des contraintes liant ces variables. Dans une telle approche, la stratégie d'affectation des variables est fortement guidée par la structure des contraintes à satisfaire et leur état de satisfaction au cours de la résolution (cf., les techniques à base de backtrack). En effet, que le problème de satisfaction des contraintes soit représenté par des variables, des contraintes ou autres (voir chapitre 2), nous considérons que la résolution reste toujours guidée directement ou indirectement par la structure des contraintes.

Bien qu'elles soient traitées suivant deux stratégies différentes, les variables représentent donc un point central dans tout processus d'optimisation ou de satisfaction de contraintes. Ainsi, quel que soit l'algorithme de résolution adopté, une large place doit être accordée à l'étude des variables d'affectation et leur dépendance vis-à-vis des objectifs d'optimisation et des contraintes à satisfaire. Cette tâche s'annonce dores et déjà délicate et nécessite une analyse fine du rôle joué par les variables dans notre approche d'optimisation sous contraintes.

Après une présentation de la particularité de notre approche et de la stratégie d'optimisation sous contraintes que nous adoptons, nous procédons à l'étude des différents types de variables intervenant afin de dégager le mécanisme adéquat permettant d'orienter l'affectation de ces variables pour assurer simultanément les tâches d'optimisation et de satisfaction.

II.2.2. Particularité de notre approche

Dans les algorithmes classiques d'optimisation sous contraintes, le processus de résolution repose généralement sur l'affectation de valeurs à des variables simples dans le but

d'améliorer l'objectif recherché tout en satisfaisant les contraintes [LAWLER 96]. Nous distinguons deux différences essentielles avec notre approche de gestion du risque de ruissellement :

- Contrairement aux problèmes classiques manipulant un nombre restreint de variables, nous disposons dans notre approche d'un grand nombre de variables (de l'ordre de quelques centaines).

- Les contraintes, dans les approches classiques, lient généralement des variables simples entre elles. Or, dans notre approche, les contraintes lient des instances de structure (les parcelles dans notre cas applicatif), en agissant sur les variables caractérisant ces instances (occupation des parcelles, sensibilité, …).

Nous distinguons dans notre approche trois couches liées à l'expression des contraintes, instances et variables simples (figure 37). Ces trois couches réparties selon deux points de vue : utilisateur et système, traduisent la complexité de notre approche par rapport aux approches classiques limitées généralement aux couches contraintes et variables simples.
Au niveau utilisateur, une contrainte est exprimée par une condition liant des instances de structure. Au niveau système, nous définissons une contrainte comme étant une restriction qui spécifie les valeurs que doit prendre un ensemble de variables simples caractérisant les instances liées par cette contrainte. Des conditions de sélection, spécifiant les valeurs que doivent prendre des variables simples, déterminent pour chaque contrainte le sous-ensemble d'instances qu'elle implique. Ces variables simples représentent les paramètres de sélection de la contrainte. Une contrainte mono-instance prend généralement la forme d'une condition logique, appelée condition de satisfaction, devant être vérifiée par chaque instance impliquée. Cette condition, spécifiant les valeurs que doivent prendre des variables simples, déterminent pour chaque contrainte le sous-ensemble d'instances, parmi celles impliquées, satisfaisant la contrainte. Ces variables simples représentent les paramètres de satisfaction de la contrainte. Une instance qui vérifie la condition logique de la contrainte est appelée instance satisfaisante. Dans le cas d'une contrainte multi-instances, la condition logique s'applique simultanément sur l'ensemble des instances impliquées dans la contrainte via une fonction mathématique que nous appelons fonction d'agrégation. Nous ne pouvons donc pas parler d'instances satisfaisantes (individuellement) dans ce cas.

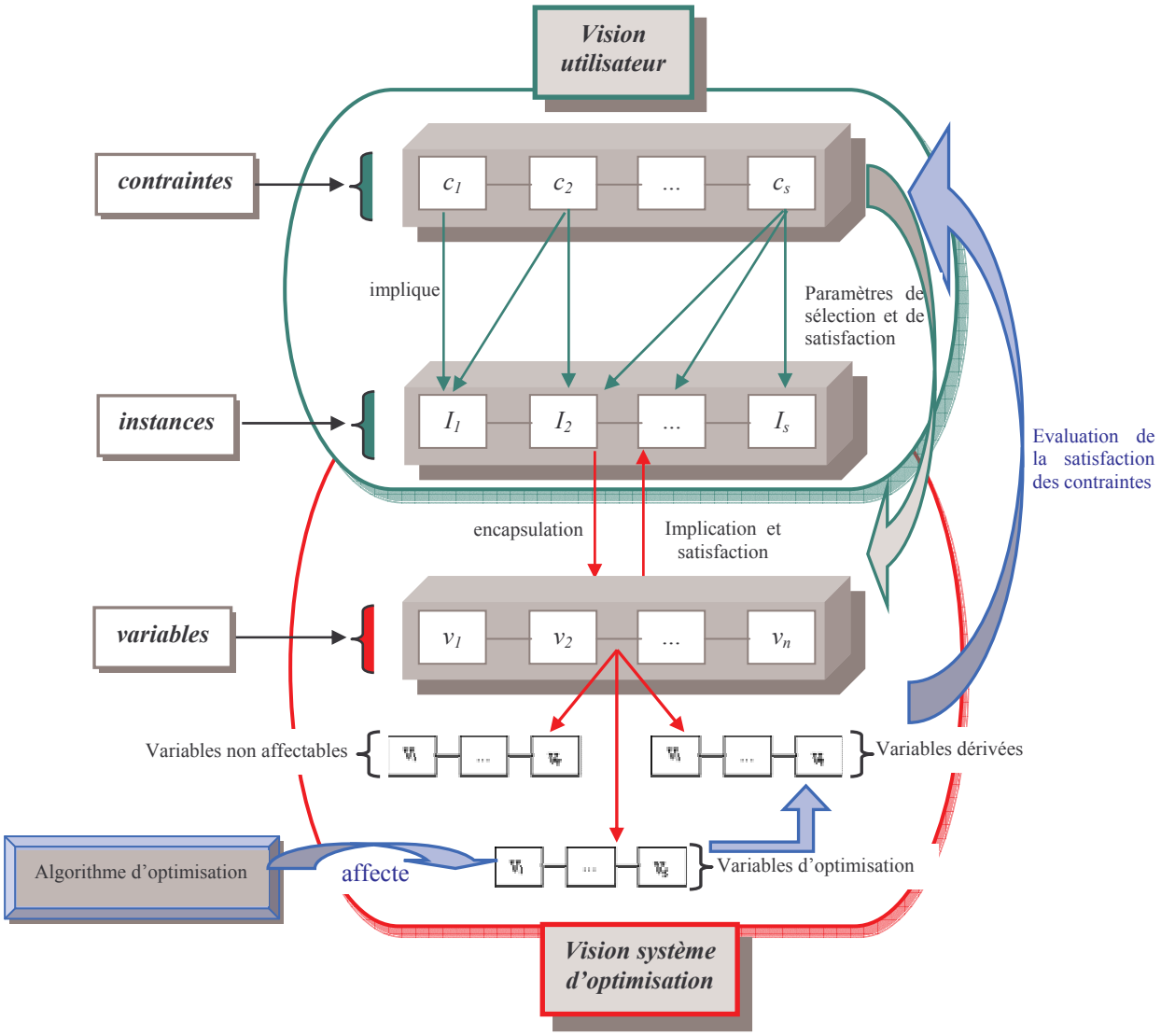


Figure 37 : Les trois couches : contraintes, instances et variables.

Au cours d'un processus d'optimisation, les valeurs de certaines variables peuvent varier d'une itération à l'autre. Les instances voient donc certains de leurs attributs changer de valeur. Ces attributs représentent les variables d'affectation. Chaque contrainte impliquant une ou plusieurs instances, la vérification de sa satisfaction nécessite la vérification de la valeur des variables caractérisant les instances impliquées dans cette même contrainte. La satisfaction des contraintes est donc susceptible de changer suite aux changements des valeurs des variables d'affectation.

Pour évaluer la satisfaction des contraintes au cours d'un processus d'optimisation, la façon la plus simple (et la plus naïve) consiste à parcourir après chaque modification des valeurs des variables d'affectation, l'ensemble de toutes les contraintes afin de vérifier si elles sont satisfaites. Or, seul un sous-ensemble d'instances voient les valeurs de leurs attributs

modifiées au cours de l'optimisation et par suite, seules les satisfactions des contraintes impliquant ces instances modifiées peuvent changer. Une façon plus judicieuse pour contrôler le niveau de satisfaction des contraintes au cours de l'optimisation consiste à ne vérifier que celles impliquant des instances modifiées. Cette démarche nécessite un passage : variables d'optimisation → instances → contraintes (figure 37). Il faut remarquer que les approches classiques d'optimisation sous contraintes abordent généralement le problème de satisfaction dans le sens : contraintes → variables. Dans une perspective d'optimisation, notre choix de passer par les variables nous oblige à inverser le schéma de données pour un passage : variables[71] → contraintes. En effet, le processus d'optimisation agit au niveau des variables d'affectation qui permettent d'identifier les instances modifiées. Les propriétés de chaque contrainte et plus précisément sa condition de sélection permet de déterminer si ces instances modifiées sont impliquées dans cette même contrainte.

II.2.3. Stratégie d'optimisation sous contraintes

Un processus classique de satisfaction des contraintes se limite généralement à la ré-affectation des variables impliquées dans des contraintes à satisfaire. A l'inverse, dans le cas de l'optimisation, toutes les variables système susceptibles d'améliorer la valeur du critère à optimiser sont généralement ré-affectées, indépendamment de leur implication dans les contraintes. Nous nous inspirons de ces deux approches pour ré-affecter les variables d'optimisation caractérisant des instances non satisfaisantes. Nous nous plaçons ici du point de vue de l'optimisation tout en restant indépendant d'un algorithme spécifique de résolution et en tenant compte des contraintes non satisfaites. Nous faisons donc abstraction des techniques spécifiques d'optimisation. Toutefois, nous orientons notre démarche vers une approche qui opère selon les concepts d'une méthode approchée car elle se prête aisément au traitement des problèmes de grande taille.

La stratégie d'optimisation sous contraintes que nous adoptons consiste à effectuer un ensemble de simulations comportant chacune une phase d'optimisation des critères pris en compte et une phase de satisfaction des contraintes exigées (voir algorithme de simulation ci-dessous).

A l'initialisation du système, une phase d'analyse des contraintes détermine les instances impliquées dans chacune. La vérification des valeurs prises par les variables simples

[71] Nous parlons ici des variables d'optimisation, indépendamment des contraintes à satisfaire. Nous ne nous limitons donc pas aux variables impliquées dans des contraintes à satisfaire.

caractérisant les instances impliquées détermine celles qui satisfont les contraintes. Une heuristique définie à ce niveau consiste à attribuer une pénalité à chaque instance qui viole une contrainte. La valeur de pénalité peut ainsi être adaptée en fonction du niveau de préférence des contraintes violées (dures, flexibles ou secondaires). L'étape d'initialisation permet donc de dégager les instances pénalisées, à l'origine de l'insatisfaction des contraintes. Le passage au niveau des variables permet de déterminer les valeurs prises par les paramètres de satisfaction caractérisant ces instances pénalisées. Ce sont ces valeurs qu'il faut modifier au cours de la résolution afin de tenter de satisfaire les contraintes non encore satisfaites. Cette modification est possible via la ré-affectation des variables d'optimisation dont dépendent les valeurs de ces paramètres de satisfaction.

La phase d'optimisation comporte plusieurs itérations correspondant chacune à la ré-affectation des variables d'optimisation caractérisant les instances pénalisées. Le processus d'optimisation favorise le traitement des instances les plus pénalisées ou encore celles violant les contraintes dures. La stratégie d'exploration des solutions est fondée sur la ré-affectation des variables d'optimisation dans le but d'améliorer les objectifs fixés tout en satisfaisant au maximum les contraintes. La satisfaction des contraintes ne devant être dégradée, les instances satisfaisant ces contraintes ne sont pas pénalisées et les variables d'affectation correspondantes ne seront pas ré-affectées afin de garantir le seuil initial de satisfaction.
En effet, bien que l'amélioration des taux de satisfaction des contraintes ne puisse être garantie, l'heuristique définie favorise la réduction du nombre d'instances pénalisées et oriente la résolution vers le maintient du seuil initial de satisfaction des contraintes.

Après chaque itération, le système procède à une phase de satisfaction pour ré-évaluer la satisfaction des contraintes suivant les nouvelles valeurs des variables ré-affectées. Cette ré-évaluation des contraintes permet de dégager les nouvelles instances non satisfaisantes et de procéder à une nouvelle itération pour ré-affecter les variables d'optimisation caractérisant ces instances.
Chaque simulation comporte donc deux niveaux de traitement :

- Un niveau micro exprimant un objectif explicite[72] relatif à un critère (risque de ruissellement dans notre cas applicatif) à optimiser au cours de la phase d'optimisation.

- Un niveau macro traduisant un objectif implicite[73] relatif à la satisfaction des contraintes à maintenir dans un certain état après chaque simulation.

[72] Objectif qui oriente la résolution.

Le principe de l'algorithme de simulation est le suivant :

Algorithme de simulation :

Début

```
/* Phase d'initialisation */
```

- Evaluer la satisfaction initiale des contraintes
- Extraire l'ensemble des instances non satisfaisantes /* l'algorithme peut se limiter aux instances violant des contraintes dures ou prendre en compte également les contraintes flexibles et secondaires */
- Attribuer une valeur de pénalité à chacune de ces instances
- **Répéter**

  ```
  /* Phase d'optimisation */
  ```

  - Appliquer une technique approchée d'optimisation
    - Ré-affecter les variables d'optimisation caractérisant les instances pénalisées
    - Evaluer la fonction objectif (critère à optimiser)

  ```
  /* Phase de satisfaction */
  ```

  - Evaluer la satisfaction des contraintes
  - Mettre à jour les instances pénalisées et leur valeur de pénalité
- **Jusqu'à** condition d'arrêt /* Etat de satisfaction générale */

Fin

Nous avons développé dans cette partie une approche d'optimisation sous contraintes, basée sur l'extraction et la pénalisation des instances à l'origine de l'insatisfaction des contraintes. Bien qu'elle soit guidée par l'évolution du critère à améliorer, la stratégie d'exploration des solutions repose sur la ré-affectation des variables d'optimisation caractérisant les instances pénalisées.

Les variables dans notre approche jouent un rôle central, aussi bien au niveau du processus d'exploration des solutions (pour la phase d'optimisation), qu'au niveau du mécanisme d'évaluation de la satisfaction des contraintes (pour la phase de satisfaction). Ainsi, il devient nécessaire d'analyser le rôle des variables et leur intervention au niveau du processus d'optimisation et de satisfaction des contraintes.

[73] Objectif pris en compte indirectement via une heuristique définie pour guider la résolution.

II.2.4. Classification des variables

Comme nous l'avons évoqué ci-dessus, l'expression des contraintes au niveau utilisateur se limite à la couche des instances (figure 37). Cependant, dans une perspective d'optimisation telle que nous l'envisageons, c'est à dire par l'utilisation d'une méthode approchée basée sur l'affectation de valeurs à des variables simples, il devient nécessaire d'externaliser les variables d'affectation caractérisant les instances.
Une instance de structure est une variable composée regroupant d'autres variables (simples ou elles-mêmes composées) et/ou des constantes. Chaque variable simple pouvant être impliquée dans une ou plusieurs contraintes, la modification de sa valeur peut influencer, directement ou via des variables dérivées, le degré de satisfaction des contraintes qui l'impliquent. Cette influence varie suivant le type de la variable et l'arité de la contrainte concernée. Le type de la variable peut être déterminé en fonction de son intervention dans le processus d'optimisation et sa dépendance vis-à-vis des contraintes.

a. L'intervention dans le processus d'optimisation :

Nous désignons par V_I l'ensemble des variables simples du problème d'optimisation sous contraintes et par ***Is*** l'ensemble des variables composées (les instances) encapsulant ces variables simples. La classification des variables simples suivant leur intervention dans le processus d'optimisation permet de distinguer :

- *Les variables d'optimisation* notées V_O ($V_O \subset V_I$) : Ce sont les variables directement affectées au cours du processus d'optimisation. Exemple : l'occupation des parcelles dans notre cas applicatif.

- *Les variables dérivées* notées V_D ($V_D \subset V_I$) : Ce sont des variables qui subissent une ré-affectation du fait de la modification des valeurs des variables d'optimisation dont elles dépendent par la structure des données. Nous utilisons le terme de variable dérivée par analogie avec le vocabulaire des modèles conceptuels en bases de données. Exemple : la sensibilité d'une parcelle dépend de son occupation et peut donc varier suite à la ré-affectation de cette dernière.

- *Les variables non affectables* notées V_C ($V_C \subset V_I$) : Ce sont les variables qui ne peuvent être modifiées ni directement ni indirectement par le processus d'optimisation du fait de la structure des données. Ces variables représentent du point de vue de l'optimisation des

constantes du système mais elles peuvent changer de valeur d'une simulation à une autre (au cours du temps métier). Exemple : le type de sol d'une parcelle.

b. La dépendance vis-à-vis des contraintes :

Nous analysons dans cette partie la dépendance des différents types de variables présentés ci-dessus vis-à-vis des contraintes prises en compte pour la satisfaction.
L'ensemble des variables dérivées et d'optimisation forme les variables d'affectation V_A. Une contrainte peut impliquer des variables affectables et/ou des variables non affectables. Pour que la satisfaction d'une contrainte puisse être contrôlable par le système d'optimisation, i.e., que la valeur de sa satisfaction puisse être modifiée au cours de l'optimisation, il faut qu'elle implique au moins une variable affectable. Dans le cas où la contrainte n'implique que des variables non affectables, sa satisfaction est considérée non contrôlable par le système et reste invariante au cours de l'optimisation.
Les variables d'optimisation déterminent la satisfaction des contraintes contrôlables, sans être nécessairement liées par ces contraintes. Les variables dérivées, si elles sont prises en compte par le système, interviennent systématiquement dans des contraintes. Par contre, nous considérons que les variables non affectables, qui ne jouent pratiquement aucun rôle dans le processus d'optimisation, prises en compte dans le système d'optimisation interviennent au moins dans une contrainte.

Par ailleurs, une contrainte est définie dans la littérature comme étant une restriction sur les valeurs qui peuvent être prises simultanément par un ensemble de variables [TSANG 93]. Plus formellement, une contrainte se caractérise par :

- *Var* (c_j) : l'ensemble des variables liées par la contrainte ;
- *Cond* (c_j) : la condition associée à la contrainte.

Cependant, cette définition qui convient bien au cadre de satisfaction de contraintes, considère seulement les variables liées par des contraintes mais celles-ci ne représentent pas la totalité des variables d'optimisation. En effet, les variables d'optimisation peuvent influencer l'état de satisfaction de certaines contraintes, dans lesquelles elles ne sont pas directement impliquées. L'implication d'une variable dans une contrainte peut être déterminée par la condition de sélection de cette contrainte.
La définition proposée dans la littérature est donc peu en rapport avec la démarche que nous adoptons et ne permet pas de répondre aux objectifs fixés par le système, à savoir l'orientation

du processus d'affectation des variables pour assurer à la fois les tâches d'optimisation et de satisfaction des contraintes. Nous pensons ainsi que la contrainte peut être exprimée de façon plus détaillée et plus adaptée aux exigences d'un système d'optimisation sous contraintes. L'orientation de la tâche d'optimisation pour tenir compte des contraintes à satisfaire passe par une re-formulation de la structure des contraintes et par l'analyse de l'évolution de leur satisfaction au cours d'un processus d'optimisation sous contraintes.

II.2.5. Re-formulation d'une contrainte dans le cadre de l'optimisation

Afin de proposer un cadre générique de formalisation et de traitement des contraintes dans le cadre d'un processus d'optimisation sans avoir à ré-évaluer l'ensemble des contraintes après chaque mise à jour des données, nous procédons à la redéfinition de la structure d'une contrainte. Une contrainte peut être caractérisée par une condition de sélection, une fonction d'agrégation (dans le cas de contrainte multi-instances), une condition de satisfaction, son arité conceptuelle et son degré de satisfaction à un instant donné.

a. Condition de sélection

La condition de sélection, notée $\text{CondSelect}(c_i)$, spécifie les conditions que les instances doivent vérifier pour être impliquées dans la contrainte $\boldsymbol{c_i}$. Les paramètres de sélection $\boldsymbol{Pi_{ci}} \subseteq \boldsymbol{V_I}$ sont des variables impliquées dans la sélection[74] d'une contrainte et dont les valeurs prises pour les instances déterminent celles ($\boldsymbol{Ii_{ci}} \subseteq \boldsymbol{I_S}$) impliquées par cette contrainte à un instant donné.

b. Fonction d'agrégation

La fonction d'agrégation : $\text{Agrégation}(c_i)$ de la contrainte c_i, qui concerne seulement les contraintes multi-instances, met en relation les instances impliquées dans la contrainte (via un paramètre de satisfaction). Sa valeur $\text{Agrégation}(c_i)_t$ dépend de celle prise par le paramètre de satisfaction (variable ou constante sur laquelle agit la fonction d'agrégation). L'impact de la ré-affectation d'une variable sur la satisfaction d'une contrainte multi-instances peut être exprimé par $\Delta\text{Agrégation}(c_i)_t$ qui représente la différence entre la valeur de $\text{Agrégation}(c_i)$ à l'instant t-1 et t :

$$\Delta\text{Agrégation}(c_i)_t = \text{Agrégation}(c_i)_t - \text{Agrégation}(c_i)_{t-1}.$$

[74] Par analogie avec les langages d'interrogation de base de données, ces variables représentent les paramètres de la primitive SELECT utilisée pour formuler une requête BD en langage SQL.

REMARQUE. — ΔAgrégation$(c_i)_t$ peut être exprimée tout simplement par la valeur du paramètre de satisfaction de l'instance $\boldsymbol{I_i}$ englobant la variable modifiée $\boldsymbol{v_i}$.

c. Condition de satisfaction

La condition de satisfaction CondSatisf(c_i) de la contrainte c_i, qui prend généralement la forme d'un opérateur et d'une valeur, détermine la condition qu'une ou plusieurs instances doivent vérifier afin que la contrainte soit satisfaite. Dans le cas d'une contrainte mono-instance, CondSatisf(c_i) permet d'identifier les instances $\boldsymbol{Is_{ci}} \subseteq \boldsymbol{Ii_{ci}}$ satisfaisant cette contrainte. Dans le cas multi-instances, elle spécifie la valeur que la variable d'agrégation, liant l'ensemble des instances impliquées dans la contrainte, doit prendre afin que la contrainte soit satisfaite. Les paramètres de satisfaction $\boldsymbol{Ps_{ci}} \subseteq \boldsymbol{V_I}$ sont des variables impliquées dans la condition de satisfaction d'une contrainte[75]. Dans le cas d'une contrainte mono-instance, les valeurs prises par les paramètres de satisfaction déterminent les instances, parmi celles impliquées, satisfaisant la contrainte à un instant donné. Dans le cas d'une contrainte multi-instances, ces valeurs définissent la valeur prise par la variable d'agrégation.

d. Arité conceptuelle

L'arité conceptuelle Arité(c_i) : mono ou multi-instances (indépendante du temps) détermine le mécanisme d'évaluation de la satisfaction de la contrainte c_i (une ou plusieurs instances à la fois). Nous désignons par $\Omega(Ii_{ci})_t$ le nombre effectif d'instances (dépendant du temps) impliquées dans la contrainte c_i à un instant t et par $\Omega(Is_{ci})_t$ le nombre d'instances satisfaisant cette contrainte au même instant.

e. Degré de satisfaction

Nous désignons par Satisf$(c_i)_t$, la valeur de satisfaction de la contrainte c_i à un instant t. Généralement, une contrainte est soit satisfaite, soit violée. Nous raffinons cet ordre en considérant qu'une contrainte peut être satisfaite à un certain degré[76].

Dans le cas d'une contrainte mono-instance, ce degré de satisfaction peut être déterminé mathématiquement par le rapport nombre d'instances $\Omega(Is_{ci})_t$ satisfaisant la contrainte sur le nombre d'instances $\Omega(Ii_{ci})_t$ impliquées dans la contrainte :

$$\text{Satisf}(c_i)_t = \Omega(Is_{ci})_t / \Omega(Ii_{ci})_t.$$

[75] Ces variables représentent les paramètres de la primitive WHERE utilisée pour formuler une requête SQL.
[76] Nous retrouvons les principes des CSPs flous [ROSENFELD 76].

Dans le cas d'une contrainte multi-instances, $Satisf(c_i)_t$ peut être exprimée par :

$Satisf(c_i)_t = Agrégation(c_i)_t$ / *Objectif*(c_i).

Objectif(c_i) représente la valeur spécifiée dans la condition de satisfaction $CondSatisf(c_i)$ et qui désigne l'objectif que $Agrégation(c_i)_t$ doit atteindre pour que la contrainte c_i soit satisfaite à 100%.

Illustrons par deux exemples les concepts définis dans cette partie :

Exemple 1. Soit la contrainte multi-instances : La somme des surfaces des parcelles contenant le lin doit être supérieure à 10 hectares. 'Somme' exprime ici la fonction d'agrégation et la valeur qu'elle renvoie à un instant t (par exemple 7 hectares) représente la valeur d'agrégation ($Agrégation(c_i)$ = 'Somme' et $Agrégation(c_i)_t = 7$). La condition de sélection est 'contenant le lin' (parcelle.occupation='lin'). Les paramètres de sélection et de satisfaction sont respectivement : 'occupation' et 'surface' de la parcelle. L'objectif *Objectif*(c_i) = 10 hectares. Le degré de satisfaction : $Satisf(c_i)_t = Agrégation(c_i)_t$ / *Objectif*(c_i) =7 /10.

Exemple 2. Soit la contrainte mono-instance : Toutes les parcelles sensibles doivent contenir de l'herbe. La condition de sélection est d'être une parcelle sensible ($CondSelest(c_i)$: 'parcelle.sensible ='vrai''). La condition 'contenir de l'herbe' détermine la satisfaction de la contrainte par une parcelle donnée ($CondSatisf(c_i)$: 'parcelle.occupation ='herbe''). Le degré de satisfaction : $Satisf(c_i)_t = \Omega(Is_{ci})_t / \Omega(Ii_{ci})_t$ = nombre de parcelles, parmi celles sensibles, contenant l'herbe /nombre de parcelles sensibles.

La re-formulation de la structure de la contrainte telle que nous venons de définir, permet de déterminer l'intervention d'une variable dans des contraintes et d'évaluer l'impact de la ré-affectation des variables d'optimisation sur la satisfaction de ces contraintes.

II.2.6. Intervention d'une variable dans une contrainte

Nous considérons qu'une variable $\mathbf{v_k}$ intervient dans une contrainte si elle fait partie des paramètres de sélection ou de satisfaction : $v_k \in Pi_{ci} \cup Ps_{ci}$. Nous supposons également qu'une variable ne peut appartenir à la fois à la condition de sélection et de satisfaction de la contrainte : $Pi_{ci} \cap Ps_{ci} = \phi$.

Si la variable fait partie des paramètres de sélection ($v_k \in Pi_{ci}$) alors elle détermine l'implication (ou non) de l'instance I_k correspondante dans la contrainte c_i. Par contre, si la

variable fait partie des paramètres de satisfaction ($v_k \in Ps_{ci}$) alors elle détermine si l'instance I_k satisfait la contrainte c_i.

Nous désignons par $EtatImplication(I_j, c_i)_t$, l'état d'implication (booléen vrai ou faux) de l'instance $\boldsymbol{I_j}$ dans une contrainte $\boldsymbol{c_i}$ à l'instant t, c'est à dire si elle vérifie la condition de sélection $CondSelect(c_i)$ de la contrainte.
Nous désignons par $EtatSatisfaction(I_j, c_i)_t$, l'état de satisfaction (booléen vrai ou faux) d'une contrainte mono-instance $\boldsymbol{c_i}$ par l'instance $\boldsymbol{I_j}$ à l'instant t, i.e., la vérification de $CondSatisf(c_i)$ par l'instance $\boldsymbol{I_j}$. Dans le cas d'une contrainte multi-instances, nous ne pouvons pas parler de l'état de satisfaction de cette contrainte par une seule instance.

La vérification de la satisfaction d'une contrainte après l'affectation d'une variable d'optimisation $\boldsymbol{v_o}$ suit donc le processus suivant :

- Si la variable d'optimisation affectée $\boldsymbol{v_o}$ n'intervient pas directement ou indirectement (via les variables dérivées) dans la condition de sélection ou de satisfaction de la contrainte alors l'affectation de la variable au cours de l'optimisation n'a aucun effet sur la satisfaction de la contrainte.

- Si la variable affectée $\boldsymbol{v_o}$ est la seule variable impliquée dans la condition de satisfaction de la contrainte ($Ps_{ci} = \{\boldsymbol{v_o}\}$), la vérification de la satisfaction de la contrainte se fait directement en vérifiant la valeur de la variable $\boldsymbol{v_o}$.

- Dans les autres cas, d'autres variables (qui dépendent de $\boldsymbol{v_o}$) interviennent dans la condition de sélection ou de satisfaction de la contrainte. La vérification de la satisfaction de la contrainte passe par la vérification des variables caractérisant $\boldsymbol{I_o}$ encapsulant la variable affectée $\boldsymbol{v_o}$.

II.2.7. Impact de l'optimisation sur le degré de satisfaction des contraintes

Nous cherchons dans cette partie à exploiter la forte dépendance entre les variables et les contraintes et leur complémentarité pour mieux assurer simultanément les deux tâches d'optimisation et de satisfaction. La re-formalisation de la structure des contraintes telle que nous venons de l'évoquer permet de mesurer l'effet de la modification des données du problème, et plus précisément les valeurs des variables d'affectation sur l'évolution du degré de satisfaction des contraintes au cours de l'optimisation. Cet effet dépend de la stratégie de

résolution adoptée, de l'arité conceptuelle des contraintes et des variables impliquées dans chacune.

Nous analysons dans cette partie l'impact de la ré-affectation d'une variable v_o suivant l'arité conceptuelle mono ou multi-instances de la contrainte. La valeur de satisfaction d'une contrainte mono-instance reflète sa satisfaction par chaque instance impliquée. La valeur de satisfaction d'une contrainte multi-instances quant à elle dépend de la fonction d'agrégation qui met en relation les instances impliquées dans cette contrainte.

a. Cas des contraintes mono-instance (intra ou inter-entités) :

La vérification de la satisfaction de la contrainte se fait pour une instance à la fois et successivement pour toutes les instances impliquées dans cette contrainte à l'instant considéré. Une contrainte mono-instance ne comporte pas de fonction mathématique (e.g., somme, produit). L'impact de la modification d'une instance (par la ré-affectation d'une variable qui la caractérise) sur la contrainte mono-instance peut être évalué en testant si cette instance vérifie la contrainte avant et après la modification. Comme nous l'avons déjà évoqué, la valeur de satisfaction d'une contrainte mono-instance peut être exprimée par le rapport nombre d'instances satisfaisant la contrainte sur le nombre total d'instances impliquées : $Satisf(c_i)_t = \Omega(Is_{ci})_t \ / \ \Omega(Ii_{ci})_t$. La contrainte est satisfaite à 100% si toutes les instances impliquées satisfont la contrainte, i.e., $Is_{ci} = Ii_{ci}$.

L'impact de la modification de la valeur d'une variable v_o sur la contrainte affecte le nombre d'instances impliquées ($\Omega(Ii_{ci})_t$) dans la contrainte, le nombre d'instances satisfaisant cette contrainte ($\Omega(Is_{ci})_t$) et par conséquent le taux de satisfaction ($Satisf(c_i)_t$). La modification de la satisfaction d'une contrainte mono-instance prend donc, en général, la forme d'une incrémentation ou décrémentation de $\Omega(Is_{ci})_t$ et $\Omega(Ii_{ci})_t$:

- Le nombre d'instances impliquées dans la contrainte ($\Omega(Ii_{ci})_t$) est modifié si l'état d'implication de l'instance dans la contrainte avant et après la ré-affectation de la variable v_o subit un changement, i.e., l'instance concernée était impliquée dans la contrainte et ne l'est plus (décrémentation de $\Omega(Ii_{ci})_t$) ou si elle n'était pas impliquée et que la modification de la valeur d'une de ses variables a entraîné son implication dans la contrainte (incrémentation de $\Omega(Ii_{ci})_t$).

- Le nombre $\Omega(Is_{ci})_t$ d'instances satisfaisant la contrainte, parmi celles impliquées, est modifié si l'état de satisfaction de la contrainte par l'instance concernée avant et après la ré-

affectation de la variable v_o change. L'état de satisfaction d'une contrainte par une instance I_o change dans les situations suivantes :

- Si I_o était impliquée dans la contrainte c_i à l'instant t-1 en la satisfaisant et elle n'est plus impliquée à t : décrémentation de $\Omega(Is_{ci})_t$.

- Si I_o est impliquée dans la contrainte à t-1 et t mais elle satisfaisait c_i à l'instant t-1 et ne la satisfait plus à t : décrémentation de $\Omega(Is_{ci})_t$.

- Si I_o n'était pas impliquée dans la contrainte à l'instant t-1 et elle est impliquée à t en la satisfaisant : incrémentation de $\Omega(Is_{ci})_t$.

- Si I_o est impliquée dans la contrainte à t-1 et t, elle ne satisfaisait pas c_i à l'instant t-1 et elle la satisfait à t : incrémentation de $\Omega(Is_{ci})_t$.

b. Cas des contraintes multi-instances (intra ou inter-entités) :

La contrainte comporte une fonction mathématique liant l'ensemble des instances qu'elle implique. Le nombre d'instances impliquées et celui d'instances satisfaisant la contrainte n'est pas significatif puisque nous ne pouvons pas parler de la satisfaction d'une contrainte multi-instances par une seule instance à la fois[77]. L'évaluation de la satisfaction de la contrainte nécessite la vérification simultanée de toutes les instances impliquées.
Comme nous l'avons déjà évoqué, nous exprimons la satisfaction d'une contrainte multi-instances par le rapport : valeur de la fonction d'agrégation sur l'objectif à atteindre pour satisfaire la contrainte : $Satisf(ci)_t = Agrégation(ci)_t / Objectif(ci)$.

L'impact d'une modification de la valeur d'une variable v_o sur la satisfaction d'une contrainte dépend donc seulement de l'état d'implication de l'instance I_o (encapsulant v_o) avant et après la modification. Si I_o n'était pas impliquée dans la contrainte avant la ré-affectation de v_o et son état d'implication reste invariant après la ré-affectation, alors aucune modification sur la satisfaction : $Satisf(ci)_t = Satisf(ci)_{t-1}$. Autrement, la détermination de l'impact de la modification de la satisfaction de la contrainte passe par la prise en compte de la valeur du paramètre de satisfaction de la contrainte, suivant la fonction d'agrégation exprimant l'aspect multi-instances pour la même contrainte (somme, nombre, moyenne). L'affectation d'une variable a un impact sur la valeur de cette fonction mathématique ($Agrégation(ci)_t$) qui détermine si la contrainte est satisfaite.

[77] A l'exception, bien entendu, du cas multi-instances avec $\Omega(Ii_{ci})_t = 1$.

Exemple : Si la fonction mathématique exprimant l'aspect multi-instances dans la contrainte est somme, il s'agit de retirer ou d'ajouter la valeur du paramètre de satisfaction.

Algorithme de suivi de l'évolution de la satisfaction des contraintes au cours de l'optimisation.

Objectif : Mesurer l'impact de la modification de la valeur d'une variable d'optimisation $\boldsymbol{v_o}$ ou d'une variable v_j dérivée caractérisant une instance $\boldsymbol{I_j}$, sur la satisfaction d'une contrainte $\boldsymbol{c_i}$.

Début

Si ni v_o ni aucune autre variable dérivée dépendant de v_o, n'intervient dans la contrainte $\boldsymbol{c_i}$ ***alors***

Aucune implication sur la contrainte quelle que soit son arité conceptuelle et la variable $\boldsymbol{v_o}$ affectée.

Sinon (v_o ou une variable dérivée v_j dépendant de v_o intervient dans la contrainte $\boldsymbol{c_i}$)

Si v_j intervient dans la condition de sélection : $v_j \in \boldsymbol{Pi_{ci}}$ ***alors***

Si $\boldsymbol{c_i}$ est mono-instance ***alors***

Vérification et mise à jour de $\boldsymbol{Ii_{ci}}$ et $\Omega(\mathrm{Ii_{ci}})_t$.

Vérification et mise à jour de $\boldsymbol{Is_{ci}}$ et $\Omega(\mathrm{Is_{ci}})_t$.

Sinon ($\boldsymbol{c_i}$ est multi-instances)

Vérification et mise à jour de $\boldsymbol{Ii_{ci}}$ et $\Omega(\mathrm{Ii_{ci}})_t$.

Mise à jour de $\mathrm{Agrégation}(c_i)_t$ et $\mathrm{Satisf}(c_i)_t$.

finSi

Sinon (v_j intervient dans la condition de satisfaction : $v_j \in \boldsymbol{Ps_{ci}}$)

// L'implication de $\boldsymbol{I_j}$ dans $\boldsymbol{c_i}$ avant et après la ré-affectation de $\boldsymbol{v_o}$ reste invariante $\mathrm{EtatImplication}(I_j, c_i)_t = \mathrm{EtatImplication}(I_j, c_i)_{t-1}$ et par suite $\boldsymbol{Ii_{ci}}$ est invariant.

Si c_i est mono-instance ***alors***

Si $\mathrm{EtatImplication}(I_j, c_i)_{t-1}$ = vrai ***alors // $\boldsymbol{I_j}$*** était impliquée dans $\boldsymbol{c_i}$ à t-1

Vérification et Mise à jour de $\boldsymbol{Is_{ci}}$ et $\mathrm{Satisf}(c_i)_t$.

Sinon ($\mathrm{EtatImplication}(I_j, c_i)_{t-1}$ = faux) ***alors***

$\mathrm{Satisf}(c_i)_t = \mathrm{Satisf}(c_i)_{t-1}$ // La satisfaction est invariante

finSi

Sinon (c_i est multi-instances)

Si $\mathrm{EtatImplication}(I_j, c_i)_{t-1}$ = vrai ***alors // $\boldsymbol{I_j}$*** était impliquée dans $\boldsymbol{c_i}$ à t-1

Vérification et Mise à jour de $\mathrm{Agrégation}(c_i)_t$ et $\mathrm{Satisf}(c_i)_t$.

Sinon ($\mathrm{EtatImplication}(I_j, c_i)_{t-1}$ = faux) ***alors***

/* Aucun impact puisque l'état d'implication de I_j est invariant */

$\mathrm{Agrégation}(c_i)_t = \mathrm{Agrégation}(c_i)_{t-1}$ // La valeur d'agrégation est invariante

$\mathrm{Satisf}(c_i)_t = \mathrm{Satisf}(c_i)_{t-1}$ // La satisfaction est invariante

finSi

finSi

finSi

Fin

III. Conclusion

Nous avons traité dans ce chapitre un aspect dynamique comportant deux étapes :

- La négociation des contraintes entre le système et le décideur au moyen d'opérations primitives et complexes de manipulation des contraintes exigées par les décideurs.

- L'analyse de la complexité totale d'un processus d'optimisation via l'étude de la complexité des contraintes en se basant sur le paramètre arité conceptuelle ainsi que la formalisation et le suivi de l'évolution de la satisfaction des contraintes au cours d'un processus d'optimisation.

Une solution à notre problème consiste à affecter au mieux les cultures sur les parcelles du territoire afin de minimiser le risque collectif de ruissellement tout en respectant les contraintes individuelles de chaque décideur. Le choix de la méthode doit assurer un certain compromis issu de la contradiction entre la complexité des contraintes exigées par les décideurs et l'obligation de minimiser la valeur du risque à l'exutoire du bassin versant.
Le chapitre suivant présente la structure de notre système de simulation développé et son application dans le cadre d'une expérimentation sur un cas réel de gestion de risque de ruissellement touchant la région du Pays de Caux au nord-ouest de la France.

CHAPITRE 5 :

Système interactif d'optimisation sous contraintes :

Application à la gestion du risque de ruissellement

Sommaire

Nous avons développé tout au long de ce travail des mécanismes d'analyse et de traitement des contraintes pour les problèmes d'optimisation. Nous tentons dans ce chapitre de projeter les concepts développés à travers une expérimentation autour des problèmes de lutte contre les risques de ruissellement touchant les territoires agricoles.

Après une introduction rappelant la problématique applicative, nous présentons l'architecture générale du système de simulation développé pour la résolution interactive des problèmes d'optimisation sous contraintes. Une modélisation de l'application est ensuite présentée comprenant le codage du problème traité en CSOP et la liste des contraintes applicatives à satisfaire par le système de simulation. Nous proposons enfin deux expérimentations effectuées dans la région du Pays de Caux et plus particulièrement sur les bassins versants de Villers-Ecalles et de Grainville-la-Teinturière avant de conclure avec une synthèse ce chapitre applicatif.

I. Introduction

Comme nous l'avons vu au chapitre 1, notre travail s'intéresse à la gestion des risques naturels touchant les territoires agricoles. Nous nous plaçons dans le nord-ouest de la France, où les exploitations de type polyculture-élevage sont dominantes, et où l'habitat est groupé. Le Pays de Caux, situé en Seine Maritime, fait partie des régions gravement touchées par les processus d'inondations et de ruissellement [DELAHAYE 95] qui motivent notre étude. Les équipes de géographes et d'agronomes sont mobilisées pour étudier les caractéristiques agronomiques et hydrologiques de cette zone ainsi que les profils financiers et comportementaux des agriculteurs.

La zone d'étude est composée en majorité de sols limoneux de granulométrie très fine (taille des particules) et donc sensibles à l'action ruisselante et érosive des pluies et des travaux de cultures. Ces sols comportent des taux très faibles de matière organique qui joue un rôle essentiel dans la protection du sol en assurant sa cohésion et sa cimentation. Les pentes sont relativement faibles (moins de 5%) ce qui favorise la mise en culture des terres. L'analyse de la couverture du sol démontre une prédominance des surfaces cultivées qui représentent 63% de la surface totale de la zone d'étude (village compris) [DELAHAYE 95].

Nous présentons dans cette partie la nécessité d'un outil de simulation de risque et le fonctionnement de l'automate cellulaire développé par l'équipe de géographes pour modéliser la dynamique des écoulements et évaluer la valeur du risque dans un bassin versant.

I.1. Nécessité d'un outil de simulation de risque

Dans le cadre de notre application, nous cherchons à minimiser la valeur du risque en ré-affectant les cultures sur les parcelles du bassin versant. A cet effet, il est important de disposer d'un outil permettant de localiser et de chiffrer avec une relative précision, la quantité de ruissellement et d'érosion vis à vis d'une certaine répartition des cultures sur les parcelles exploitées.
La modélisation des processus de risque et la cartographie des zones sensibles ont fait l'objet de nombreux travaux [CERDAN 01][DE ROO 93][KING 92][LARDON 92] [SOUCHERE 95]. Cependant, la plupart des méthodes proposées sont lourdes d'utilisation en raison d'un important besoin de données agronomiques d'entrée ou ne permettent pas de mesurer l'incidence de la position d'une surface contributive sur le bassin versant [LANGLOIS 02]. Elles sont donc peu en rapport avec nos objectifs cherchant à intégrer, pour chaque zone, sa dynamique interne et également ses relations avec son environnement spatial afin d'évaluer sa réaction aux apports venus de l'amont et ses répercussions sur les parties avales.

I.2. Automate cellulaire de simulation du risque

L'équipe des géographes développe un outil de simulation "RuiCells" [LANGLOIS 02], à base d'automate cellulaire, qui modélise la dynamique des écoulements tout au long du bassin versant. Il permet de gérer à la fois la morphologie du terrain, la diversité de l'occupation du sol et la dynamique des écoulements entre les différentes parcelles du territoire. La structure de l'automate est formée de cellules surfaciques, linéaires et ponctuelles reliées par un graphe d'écoulement. Sa construction nécessite un maillage régulier ou une triangulation de Delaunay du domaine d'étude. L'espace est ainsi découpé en cellules homogènes connectées entre elles. Avec un Modèle Numérique de Terrain (M.N.T) au pas de 75 mètres, une cellule représente une surface de 3000 m² environ. Nous renvoyons à l'annexe 3 pour de plus amples détails sur le système de simulation utilisé à ce niveau.

L'automate cellulaire permet de déterminer des indices de sensibilité aux inondations au sein d'un bassin et de mesurer, par simulation, la valeur du ruissellement cumulé en un point

donné du territoire suite à un événement pluvieux. C'est cette activité de ruissellement que nous cherchons à minimiser à l'exutoire du bassin versant, en modifiant la distribution des cultures sur le parcellaire.
L'évaluation de la valeur du risque par l'automate cellulaire étant au niveau des éléments de surface (cellules), pour proposer une décision d'assolement, nous devons mettre en œuvre un raisonnement à l'échelle de l'unité décisionnelle (le parcellaire). Une parcelle couvre généralement entre 1 et 30 hectares et peut regrouper jusqu'à 100 cellules. La simulation permet de caractériser l'activité de ruissellement de chaque élément de surface. Il est ainsi possible de caractériser cette activité pour chaque parcelle en intégrant l'activité de l'ensemble des cellules qui la composent.

II. Architecture du système de simulation

Cette partie est consacrée au système de simulation que nous avons développé pour la résolution des problèmes d'optimisation sous contraintes. Nous présentons le fonctionnement global du système avant de détailler ses quatre phases de conception qui synthétisent les concepts développés tout au long de ce travail.

II.1. Fonctionnement du système

Afin de mettre en oeuvre les idées exposées tout au long de ce travail, nous avons développé un système de simulation intégrant des phases d'optimisation, de satisfaction de contraintes et d'aide à la décision à travers une démarche interactive de progression dans la saisie, la négociation et la satisfaction des besoins des décideurs. Nous espérons ainsi sensibiliser les décideurs individuels aux problèmes collectifs et les amener à prendre en compte le risque de ruissellement dans leurs pratiques agricoles. La figure 38 donne la structure logique du système de simulation développé.

En reprenant la démarche retenue dans la modélisation du problème (présentée dans la partie I du chapitre 3), nous mettons en œuvre les concepts développés tout au long de ce travail pour la conception d'un système de simulation basé sur une démarche progressive et générique intégrant les décideurs dans les différentes phases de résolution. Les décideurs définissent eux-mêmes leurs besoins (partie III du chapitre 3) suivant une syntaxe bien définie, conformément à une grammaire spécifiée (partie I.1 du chapitre 4). Cependant, les contraintes exigées sont souvent complexes et difficiles à satisfaire simultanément. A cet

effet, le système intègre une phase d'interaction avec les décideurs pour impliquer ces derniers dans la négociation de leurs contraintes afin de retenir un sous-ensemble de contraintes de complexité raisonnable pouvant être traitée par le système (partie I.3 du chapitre 4).

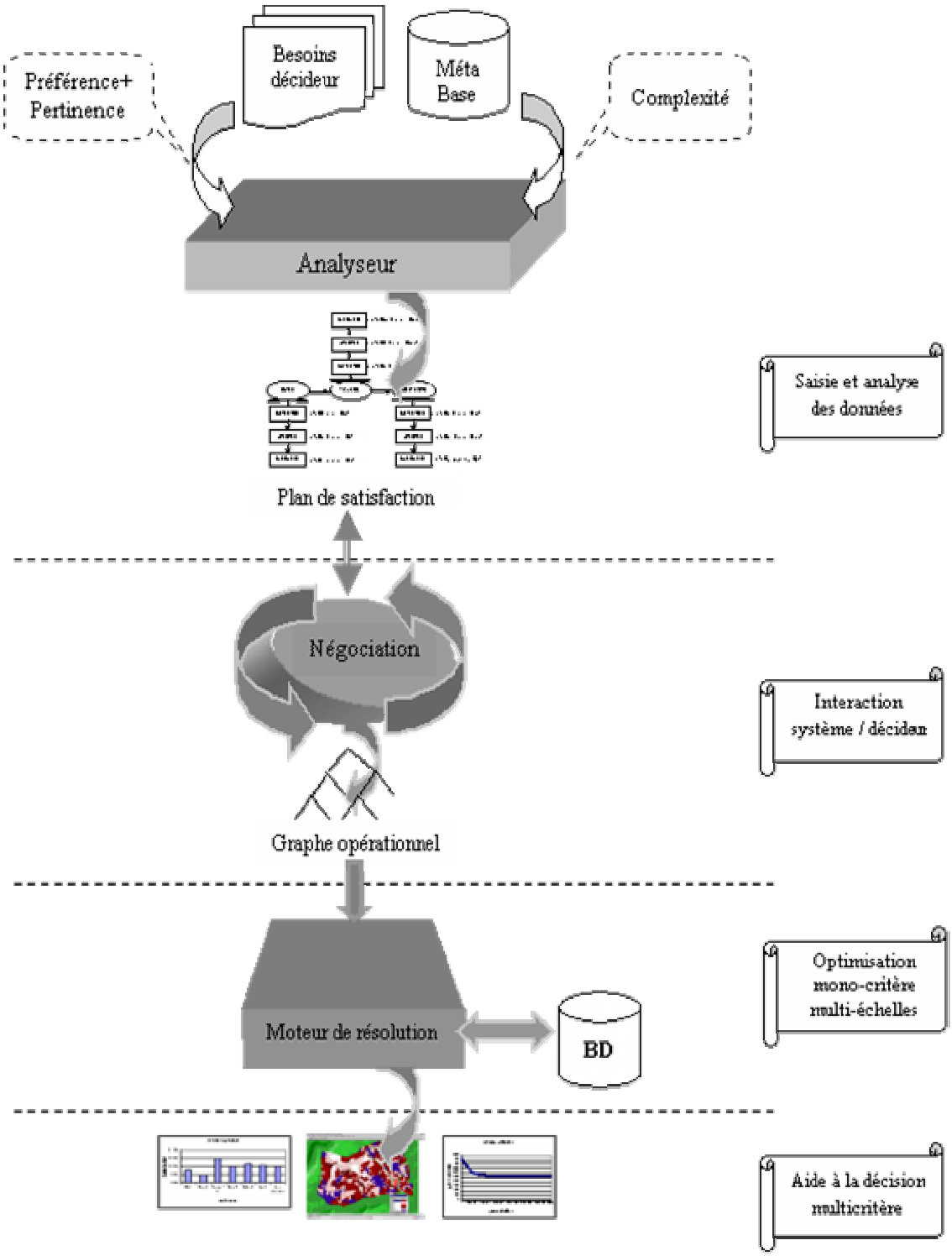


Figure 38 : Architecture logique du système de simulation.

Le processus de résolution cherche à assurer simultanément les objectifs d'optimisation et de satisfaction des contraintes à travers une démarche mono-critère multi-échelles. Il intègre ainsi l'amélioration de la fonction objectif au cours de la phase d'optimisation et le respect de la satisfaction des contraintes à l'issue de chaque simulation (partie II.2 du chapitre 4). Etant donnée la divergence entre les préférences et les évaluations des décideurs impliqués dans la prise de décision, plusieurs solutions sont proposées, définissant ainsi un cadre multicritère d'aide à la décision.

II.2. Conception du système

La conception du système peut se décomposer en quatre phases essentielles, à savoir la saisie et l'analyse des données, l'interaction entre le système et le décideur, l'optimisation multi-échelles et l'aide à la décision multicritère. Nous décrivons dans cette partie chacune des phases du processus de conception :

II.2.1. Saisie et analyse des données

Cette étape concerne la saisie des données nécessaires au fonctionnement du système de simulation et leur traitement afin d'établir une liste ordonnée de contraintes. Les données représentant les entrées du système sont saisies par les différents acteurs intervenant dans le processus, à savoir les décideurs agriculteurs, agronomes et instances locales ainsi que le concepteur de l'application. Trois grammaires sont ainsi définies afin de permettre la saisie de ces données conformément au langage spécifié par chacune[78].

- Les décideurs agriculteurs et instances locales introduisent leurs besoins exprimant les contraintes à prendre en compte et leurs préférences sur ces contraintes (partie I.2 du chapitre 4).

Exemple : La saisie de la contrainte pente qui exige que le lin dans l'exploitation 'Leber' soit placé sur des parcelles de pente < 10, prend la forme suivante :

```
# Contrainte de pente
####################
Contrainte : pente
Commentaire : " exigence de pente "
```

[78] Nous avons utilisé l'analyseur ANTLR (Another Tool for Language Recognition). ANTLR est un générateur d'analyseurs lexicaux et syntaxiques qui permet de construire des compilateurs ou des convertisseurs de code à partir de descriptions syntaxiques contenant des actions en Java ou C++. Voir site : www.antlr.org.

```
Type : Alphanumerique
Preference : Dure
Poids : 0,1
Entite : Parcelle [Occupation, Exploitation,]
Requete : Pente [Parcelle.Occupation = 'lin' et
Parcelle.Exploitation = 'leber'] < 10 ;
```

- Les experts agronomes introduisent les données métier relatives aux familles de contraintes et leur coefficient métier. Ces données permettent de déduire la pertinence métier de chaque contrainte décideur (partie I.2 du chapitre 4).

Exemple : La saisie de la famille des contraintes spatiales prend la forme suivante :

```
# Famille Métier
################
Famille : Spatiale
Coefficient metier : 30%
```

- Le concepteur saisit le modèle logique de données ainsi que les données concrètes du terrain étudié. Le modèle logique de données sert à déduire la complexité des contraintes introduites par les décideurs (partie II.1 du chapitre 4).

Exemple : La saisie de l'entité parcelle prend la forme suivante :

```
# Entite Parcelle
################
Entite   : "Parcelle"
Attribut : "Numero",                    Entier, Cle,
Attribut : "Pente",                     Reel
Attribut : "Taille",                    Entier
Attribut : "TypeSol",                   Chaine
Attribut : "Exploitation",              CleEtrangere [Exploitation]
Attribut : "Proprietaire",              CleEtrangere [Exploitant]
Attribut : "Occupation",                CleEtrangere [Occupation]
Attribut : "DistanceFerme",             Reel
Attribut : "ProfilFinancier",           Chaine
Attribut : "AttachementProprietaire",   Chaine
```

Cette phase permet de récupérer un ensemble de contraintes non ordonnées et de déduire les trois paramètres : métier (pertinence métier), décideur (niveau de préférence) et système (complexité). L'ordonnancement des contraintes dans une structure de liste (partie I.2 du

chapitre 4) se fait selon ces trois paramètres déduits à partir des données saisies par les différents acteurs et permet d'avoir le plan de satisfaction des contraintes.

II.2.2. Interaction système / décideur

A partir de la première liste de contraintes, une phase d'interaction entre le système et le décideur a pour objectif de négocier le plan de satisfaction des contraintes afin d'équilibrer le rapport temps de réponse exigé par le décideur/temps nécessaire pour la satisfaction. Un ensemble d'opérations primitives et complexes est mis à la disposition des décideurs afin de mettre à jour le système de contraintes (partie I.3 du chapitre 4). Le graphe opérationnel de traitement retenu comporte les contraintes effectives à prendre en compte par le système dans le processus de simulation afin d'offrir des solutions satisfaisantes aux décideurs.

II.2.3. Optimisation mono-critère multi-échelles

En se basant sur le vocabulaire d'échange d'informations (partie III, chapitre 3), il s'avère évident que notre problème de risque s'inscrit dans un cadre multicritère, comportant un critère de satisfaction des contraintes, le temps de réponse, le risque et la répartition spatiale des cultures[79]. La démarche suivie pour l'optimisation sous contraintes (voir partie II.2. du chapitre 4) nous a permis de simplifier ce problème d'optimisation naturellement multicritère en le ramenant à un problème mono-critère multi-échelles. Nous distinguons ainsi l'optimisation du risque à l'échelle du processus d'optimisation et la garantie d'un seuil de satisfaction des contraintes à l'échelle de la simulation. Les critères de temps de réponse et de la répartition spatiale sont pris en compte sous forme de contraintes supplémentaires.

Le processus de résolution effectue un ensemble de simulations comportant chacune une phase d'optimisation des objectifs et une phase de satisfaction ayant pour but l'évaluation de la satisfaction des contraintes et l'extraction des variables violant ces contraintes. Cet ensemble de variables est traité par le processus d'optimisation qui cherche à leur ré-affecter d'autres valeurs dans le but d'améliorer les objectifs fixés. Nous nous limitons dans notre cas à un seul objectif (le risque de ruissellement) et nous traitons les autres objectifs sous forme de contraintes tout en fixant des seuils minimums pour chacun. Plusieurs solutions sont proposées au fil du temps afin de laisser le choix libre aux décideurs dans la sélection de la meilleure solution. Nous procédons ainsi à une décision multicritère que nous estimons plus adaptée à notre cadre applicatif.

[79] La répartition des cultures correspond plus à un critère de décision qu'à un critère d'optimisation.

II.2.4. Aide à la décision multicritère

A l'image des problèmes de décision faisant intervenir le facteur humain, les critères qui interviennent lors des décisions en aménagement du territoire sont de natures très diverses. La pluralité des décideurs, exprimant souvent des points de vue différents, fait que l'évaluation des différentes solutions proposées, impliquant chacune des valeurs différentes des critères, reste relative (à chaque décideur) et non absolue. De ce fait, certaines situations peuvent être caractérisées par la difficulté de désigner la meilleure solution trouvée étant donné la divergence entre les préférences et les évaluations des différents décideurs. Cette divergence d'attitude et de comportement explique cet aspect multi-évaluations et notre choix de proposer plusieurs solutions différentes et faire ainsi recours à une décision multicritère. En plus, devant la nécessité d'un compromis entre la qualité de la solution proposée et le temps de réponse exigé, l'évaluation multicritère paraît une réponse naturelle.
Les solutions proposées dans notre cas applicatif, comportent un paramètre spatial relatif à la carte de répartition des cultures permettant ainsi la vérification de la cohérence spatiale et hydrologique de la distribution des cultures sur les parcelles du territoire et des paramètres alphanumériques relatifs à la valeur du risque, les taux de satisfaction des contraintes et le temps de réponse.

III. Modélisation de l'application

La modélisation consiste à créer une représentation simplifiée d'un problème pour mieux comprendre le système étudié. La figure 39 illustre le diagramme statique de classes qui montre la structure générale des entités concernées par notre modèle de résolution et les relations pouvant exister entre elles.

Nous présentons successivement dans cette section un codage de notre problème étudié sous forme d'un CSOP, la liste des contraintes applicatives devant être prises en compte par le système de simulation ainsi que les orientations suivies pour la résolution du problème.

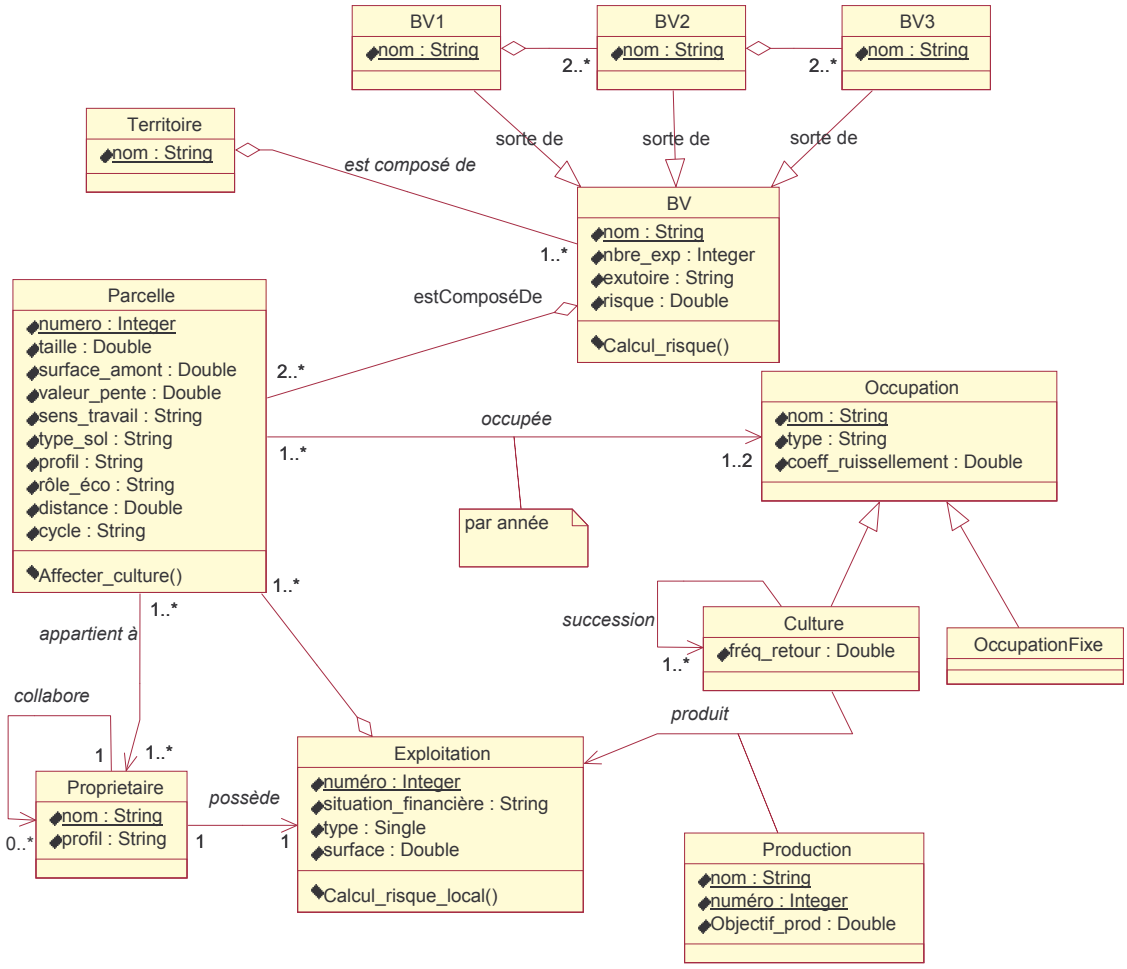


Figure 39 : Modélisation UML du diagramme statique de classes.

III.1. Codage du problème en CSOP

Nous nous intéressons dans notre travail aux problèmes d'optimisation sous contraintes. La démarche que nous suivons pour la résolution de notre problème de gestion du risque de ruissellement consiste à affecter une culture à chaque parcelle du territoire, de façon à respecter l'ensemble des contraintes du système, et réduire la valeur du risque dans l'ensemble du territoire. Une configuration correspond à une répartition des cultures sur l'ensemble des parcelles composant le bassin versant.

- ***Variables :*** Les parcelles du bassin versant représentent les variables du système.

$X = \{p_i, i \in [1, n_p], n_p$ étant le nombre de parcelles du BV$\}$.

- ***Domaines :*** Nous désignons par *D* l'ensemble des domaines finis associés aux variables.

D = {Dom (p_i) | $p_i \in X$}, Dom (p_i) = {v_j, $j \in [1, n_o]$, n_o étant le nombre d'occupations pouvant être placées sur la parcelle p_i }.

Nous distinguons deux domaines disjoints ($D = D_1 \cup D_2$) : le domaine D_1 des parcelles contenant une occupation fixe et qui se limite à une seule valeur (occupation) et D_2 celui des parcelles contenant une culture pouvant être modifiée. Nous supposons que ces parcelles ont le même domaine, i.e., toutes les cultures peuvent être placées sur ces parcelles :

- D_1 = {occupationFixe}.
- D_2 = {$culture_1$, $culture_2$, …, $culture_c$}.

- ***Contraintes*** *:* Nous désignons par C l'ensemble des contraintes du système :

C = {c_w, $w \in [1, n_s]$, n_s étant le nombre de contraintes du système}.

- ***Fonction objectif :*** Pour évaluer la performance/coût de chaque configuration, nous procédons à l'évaluation d'un paramètre spatial relatif à la carte de répartition des cultures et de trois paramètres alphanumériques relatifs à la valeur de risque, aux taux de satisfaction des contraintes et au temps de réponse. Cependant, malgré cet aspect multicritère, notre choix est porté sur l'utilisation du critère de risque, induit par chaque répartition de cultures, pour orienter la recherche de solutions au cours du processus d'optimisation. Les autres critères sont pris en compte sous forme de contraintes supplémentaires.

- ***Espace de recherche :*** Les multiples affectations possibles des cultures sur les parcelles représentent l'espace de recherche. Mathématiquement, le nombre (fini) d'affectations possibles ou de solutions potentielles (que nous désignons par l'ensemble A) est fonction du nombre n des variables et de la taille d de leur domaine. En supposant que toutes les variables ont la même taille de domaine, le nombre d'affectations possibles est de l'ordre de d^n ($A = \{A_i, i \in d^n\}$).

- ***Solutions :*** Une solution (affectation consistante) à un CSOP est une affectation de valeurs à des variables, qui satisfait les contraintes du système et qui définit la valeur de la fonction objectif. Dans notre cas multicritère, nous définissons une solution à notre problème comme étant une configuration qui vérifie un seuil minimum de satisfaction pour les contraintes exigées par les décideurs et qui améliore la valeur du risque de ruissellement tout en respectant un temps de réponse raisonnable.

III.2. Contraintes applicatives

Nous présentons dans cette partie les différentes contraintes applicatives afin d'illustrer le réalisme de notre application et la multitude de contraintes exigées par les décideurs. Ces contraintes seront ensuite classées suivant les paramètres de classification étudiés dans le chapitre 3. Le critère décideur "préférence sur les contraintes" ne sera pas pris en compte dans cette classification puisqu'il ne peut être déterminé de façon absolue et reste relatif à chaque décideur. Nous nous limitons donc à la classification des contraintes suivant les paramètres : le comportement dans le temps de simulation, l'évolution dans le temps métier et l'arité conceptuelle des contraintes.

Nous disposons dans notre application des contraintes suivantes :

1. *Exigence de pente* (contrainte statique-stable, inter-entités, mono-instance) : Cette contrainte liée à l'exigence des outils de travail apparaît en raison de l'intervention d'outils et de machines de taille et encombrement importants. Chaque culture nécessite une pente inférieure à un certain seuil au-delà duquel elle ne pourra pas être cultivée en raison de l'impossibilité d'utiliser les machines agricoles. Sur le terrain d'étude, l'herbe reste peu exigeante en terme de pente et accepte des surfaces excédant 10% de pente. La plupart des autres assolements doivent être localisées sur des pentes inférieures à 5%, à l'exception du maïs qui peut accepter des pentes allant jusqu'à 10%.

2. *Type de sol compatible* (contrainte statique-évolutive, inter-entités, mono-instance) : Le type de sol détermine les cultures qui peuvent être placées sur chaque parcelle. Un sol est limitant pour les choix d'affectation des cultures, tant du fait de ses bonnes que par ses mauvaises qualités. A ce titre, certaines cultures comme le lin, la betterave et la pomme de terre sont plus exigeantes que d'autres vis à vis du type de sol et doivent bénéficier de conditions satisfaisantes pour pouvoir pousser.

3. *Taille de parcelle compatible avec les exigences techniques et économiques* (contrainte dynamique-stable, inter-entités, mono-instance) : Bien que tout assolement[80] puisse généralement être pratiqué sur de petites surfaces, dans les systèmes de production actuels orientés cultures céréalières et industrielles, l'intervention sur de grandes parcelles répond à plusieurs impératifs techniques, agronomiques et économiques. L'intensification des modes de production, et notamment des outils et machines utilisées, nécessite de grandes surfaces

[80] A quelques exceptions près, en l'occurrence le lin et la betterave, qui exigent des possibilités de manœuvre au bord des parcelles.

pour manœuvrer. Le morcellement parcellaire multiplie les interventions sur des pièces éloignées et réparties sur le territoire de l'exploitation et ne permet pas de répondre aux objectifs de production et de productivité imposés par le système de production.

4. *Positionnement rentable des cultures sur les parcelles du bassin versant* (contrainte dynamique-stable, intra-entité, mono-instance) : Dans le cas d'une exploitation pratiquant de l'herbe et des cultures variées et disposant de terres bonnes et mauvaises, il faut éviter de positionner l'herbe (prairie) sur les bonnes parcelles situées essentiellement en amont (zones peu risquées). De la même manière, il faut éviter de placer les cultures rentables sur les mauvaises parcelles positionnées en aval du bassin (zones trop risquées) afin d'assurer un meilleur rendement des cultures.

5. *Intervalle de production autorisé* (contrainte dynamique-évolutive, inter-entités, multi-instances) : Chaque exploitant fixe un objectif de production pour chaque type de culture présente dans son exploitation. Ces objectifs doivent être respectés dans la limite du fonctionnement admissible fixé par des seuils de satisfaction déterminés par les exploitants.

6. *Favoriser les cultures les plus rentables* (contrainte dynamique-stable, inter-entités, multi-instances) : Les décideurs privilégient les cultures les plus rentables économiquement. Le rendement des cultures ne peut être exprimé de manière relative entre deux assolements, car leur utilité au sein du système d'exploitation est différente. Dans ce sens, nous distinguons les cultures de rendement économique direct (qui ont directement une signification marchande, en l'occurrence le blé) et les cultures (le maïs par exemple) qui ont un effet indirect sur l'équilibre économique de l'exploitation puisqu'elles participent entre autres, au système d'élevage en assurant la nourriture des animaux.

7. *Adaptation aux conditions du marché* (contrainte statique-évolutive, intra-entité, mono-instance) : Les systèmes de production ont tendance à se spécialiser et l'éventail des assolements pratiqués se restreint. Le système doit donc pouvoir offrir aux décideurs la possibilité de supprimer, momentanément ou définitivement, un ou plusieurs assolements.

8. *Découpage des parcelles de grande taille* (contrainte dynamique-stable, intra-entité, multi-instances) : Une parcelle de grande taille peut être découpée en plusieurs sous-parcelles (sous contraintes économiques). Le découpage parcellaire peut également s'appliquer aux blocs de jachère et il peut en effet être intéressant de multiplier de petites parcelles de jachère positionnées à des endroits très stratégiques (en terme de risque).

9. *Non-dégradation de la sensibilité de la parcelle* (contrainte dynamique-évolutive, inter-entités, mono-instance) : Lors de l'affectation des cultures, il faut éviter de placer sur une parcelle sensible, une culture plus ruisselante que celle qui y existait auparavant, puisque cette culture ne poussera pas dans ce cas.

10. *Amélioration du risque ruissellement/érosion après la simulation* (contrainte dynamique-évolutive, intra-entité, multi-instances) : Les solutions proposées doivent améliorer la valeur du risque de ruissellement pour l'ensemble du bassin versant.

11. *Sens du travail de la parcelle* (contrainte dynamique-stable, intra-entité, mono-instance) : Pour minimiser le risque, le sens de travail de la parcelle doit être perpendiculaire à la pente afin d'augmenter la résistance du sol et ralentir ainsi la vitesse d'écoulement de l'eau.

12. *Cycle d'assolement de la parcelle* (contrainte dynamique-évolutive, inter-entités, multi-instances) : L'occupation de chaque parcelle au cours du temps doit être conforme à son cycle d'assolement. Ceci permet de respecter le rythme naturel des cultures et préserver la qualité du sol pour les parcelles.

13. *Successions culturales* (contrainte statique-évolutive, inter-entités, multi-instances) : Certaines cultures ne peuvent pas se succéder dans les rotations d'assolement. Il s'agit ici de contraintes liées à la succession des cultures et à l'harmonie des cycles selon des règles agronomiques.

14. *Fréquence de retour des cultures* (contrainte dynamique-évolutive, inter-entités, mono-instance) : Au-delà des règles de succession précédent/suivant, nous devons prendre en compte la succession en terme d'historique de la parcelle. En effet, à chaque culture correspond une fréquence de retour qui détermine la période minimale qui doit séparer deux présences successives d'une culture sur la même parcelle.

15. *Temps de réponse acceptable* (contrainte dynamique-évolutive, intra-entité, mono-instance) : Etant dans un système interactif d'aide à la décision, les solutions proposées ne doivent pas dépasser un temps de réponse raisonnable.

16. *Distance à la ferme pour l'herbe et le maïs* (contrainte statique-évolutive, inter-entités, mono-instance) : Une dimension spatiale doit être prise en compte dans le cas d'une exploitation agricole avec élevage. Si l'activité d'élevage est présente dans l'exploitation et si la remise en herbe ou en maïs est proposée, la parcelle concernée ne doit pas se situer trop loin du corps de ferme qui constitue le point de référence pour l'exploitation.

17. *Taux de Jachère surfacique* (contrainte dynamique-évolutive, inter-entités, multi-instances) : Les exploitations agricoles doivent appliquer au moins 10% de jachère pour les surfaces cultivées, à partir d'un seuil de 13 hectares de cultures. Ce seuil ne se calcule que par rapport aux surfaces dites SCOP (blé, maïs, pois, colza, escourgeon).

18. *Fourrière de 20 m de côté* (contrainte statique-évolutive, intra-entité, mono-instance) : Il est préférable de placer la jachère sur les bandes appelées fourrières (en bout de parcelle) et vérifier une surface d'au moins 20 mètres de côté. Ces bandes freinent le ruissellement et augmentent la résistance du sol à l'incision.

19. *Non-réaffectation des parcelles contenant des occupations fixes* (contrainte statique-stable, intra-entité, mono-instance) : Les occupations fixes (bois et zones d'habitats) ne doivent pas être déplacées de leurs parcelles initiales.

20. *Vérification de la cohérence des requêtes utilisateurs* (contrainte statique-stable, inter-entités, multi-instances) : Le système doit contrôler les demandes des utilisateurs afin de s'assurer de leur conformité avec les connaissances du terrain et la logique de résolution. Exemple : les quantités de production demandées par un exploitant ne doivent pas dépasser la capacité de production de ses parcelles.

21. *Unité décisionnelle de la parcelle* (contrainte statique-stable, intra-entité, multi-instances) : Un exploitant peut accepter de collaborer avec un ou plusieurs autres exploitants. Cette collaboration se manifeste par des échanges de parcelles entre leurs exploitations respectives. Les permutations des parcelles appartenant à des exploitations différentes doivent donc se faire uniquement entre les exploitants collaborants.

22. *Amélioration du risque local au niveau de l'exploitation* (contrainte dynamique-évolutive, inter-entités, multi-instances) : Certains agriculteurs très touchés par les processus de ruissellement, demandent l'amélioration de la valeur du risque au sein de leur exploitation indépendamment du reste du territoire.

La figure 40 récapitule le classement des contraintes applicatives suivant la grille de correspondance métier/simulation (partie V.2 du chapitre 3).

Relaxation simulation	Evolution métier	Entité	Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Contraintes statiques																									
	Stable																								
		Intra-entité																							
			Mono-instance																			*			
			Multi-instances																					*	
		Inter-entités																							
			Mono-instance	*																					
			Multi-instances																				*		
	Evolutive																								
		Intra-entité																							
			Mono-instance							*											*				
			Multi-instances																					*	
		Inter-entités																							
			Mono-instance		*														*						
			Multi-instances													*									
Contraintes dynamiques																									
	Stable																								
		Intra-entité																							
			Mono-instance				*							*											
			Multi-instances								*														
		Inter-entités																							
			Mono-instance			*																			
			Multi-instances						*																
	Evolutive																								
		Intra-entité																							
			Mono-instance															*							
			Multi-instances								*		*												
		Inter-entités																							
			Mono-instance									*					*								
			Multi-instances					*							*					*					*

Figure 40 : Répartition des contraintes applicatives suivant la grille de correspondance métier/simulation.

Cette classification reste dépendante du point de vue du concepteur et du contexte applicatif. En effet, la même contrainte applicative peut être exprimée par plusieurs contraintes décideurs et avoir ainsi différentes classifications possibles.

Exemple : La contrainte applicative 5 relative à l'intervalle de production autorisé peut avoir les instances suivantes :

- Contrainte 1 (décideur Leber) : La production du lin dans l'exploitation Leber doit dépasser 3 hectares.
- Contrainte 2 (décideur Leber) : La production du maïs dans l'exploitation Leber ne doit pas dépasser 30% de la somme totale des productions de ses cultures.

- Contrainte 3 (décideur David) : La production du blé dans l'exploitation David ne doit pas dépasser celle des années dernières.

III.3. Orientations de résolution

Pour optimiser le placement des cultures sur les parcelles, deux approches sont possibles selon le mécanisme de résolution adopté. Nous distinguons l'approche basée sur des affectations et celle basée sur des permutations. Toutefois, ces deux approches peuvent être combinées, i.e., utiliser l'approche d'affectation pour générer une première solution et appliquer un processus de permutations pour optimiser les placements.

- *Approche basée sur des affectations* : Cette approche consiste à partir d'une occupation initiale vide et à construire une solution au problème en affectant une culture à chaque parcelle du territoire. Nous trouvons ici les concepts de l'approche de construction (présentée dans le chapitre 2). Cependant, bien qu'elle offre une grande marge de manœuvre pour le système et qu'elle permette d'orienter l'affectation vers les concepts de gestion globale du bassin versant, cette approche part d'une occupation initiale vide qui ne reflète pas la réalité observée sur le terrain. De plus, elle ne garantit pas une meilleure configuration que celle choisie initialement par les décideurs. Cette approche ne tient donc pas compte des choix initiaux effectués par les agriculteurs chacun sur ses propres parcelles. Cet aspect peut s'avérer contraignant si nous fixons pour perspective d'aider les décideurs à mieux organiser leurs espaces individuels au lieu de les gérer à leur place.

- *Approche basée sur des permutations* : Elle consiste à partir d'une occupation initiale des parcelles (qui peut éventuellement représenter l'occupation observée sur le territoire) et à effectuer des permutations entre les occupations du sol afin d'améliorer le risque tout en respectant les contraintes des décideurs.

Notre choix s'est porté sur la deuxième approche afin de considérer l'occupation initiale choisie par les agriculteurs et de partir avec une configuration acceptée par les différents décideurs.

Nous présentons dans la partie suivante deux approches de résolution basées sur un mécanisme de permutations. La première approche orientée optimisation représente une pré-étude opérant suivant les principes d'une technique approchée et vise à apporter des solutions de ré-affectation de cultures favorisant la minimisation du risque de ruissellement. Cette approche qui ne considère qu'un nombre restreint de contraintes, montre la limite des

méthodes approchées face aux problèmes d'optimisation sous contraintes. A cet effet, nous procédons à une seconde approche plus générique visant à assurer simultanément les objectifs d'optimisation et de satisfaction des contraintes. Cette approche met en œuvre le système de simulation développé en se basant sur une stratégie de résolution inspirée de l'algorithme d'optimisation sous contraintes développé (partie II.2 du chapitre 4).

IV. Première approche : Modèle expérimental d'optimisation

Une étude menée sur les problèmes de ruissellement dans le bassin versant de Villers-Ecalles (14 km² dans le Pays de Caux) a permis de concevoir un premier modèle de simulation cherchant à minimiser le risque de ruissellement tout en respectant un ensemble restreint de contraintes. L'ensemble du bassin versant est considéré comme une seule exploitation, i.e., nous autorisons la permutation des parcelles appartenant à des exploitations différentes. Nous disposons de 529 parcelles partagées entre 7 types d'occupations (blé, maïs, lin, colza, prairie, bois et zone d'habitat). Les zones boisées et d'habitat sont considérées fixes et ne sont donc pas prises en compte dans le processus de ré-affectation des occupations du sol sur les parcelles. Elles n'interviennent que lors du calcul du risque global de ruissellement à l'exutoire du bassin versant. La traduction de cette première expérimentation en CSOP prend la forme suivante :

- $X = \{p_1, p_2, ... p_{529}\}$ ensemble des 529 parcelles du territoire ;
- $D = D_1 \cup D_2$
  - D_1 = {zone d'habitat, bois} ;
  - D_2 = {blé, maïs, lin, colza, prairie} ;
- $C = \{c_1,... c_s\}$ ensemble des *s* contraintes ;
- F : Minimiser la valeur du risque de ruissellement à l'exutoire du bassin versant.

IV.1. Stratégie de résolution

En partant de l'occupation existante relevée sur le terrain, la stratégie de résolution consiste à proposer des modifications d'assolement basées sur un mécanisme de permutations de cultures entre les différentes parcelles en tenant compte du risque de ruissellement. Nous avons orienté notre choix sur un modèle d'affectation qui opère selon des principes inspirés du paradigme multi-agents car il se prête aisément à l'intégration de plusieurs niveaux de

contraintes [JAZIRI 02b]. Son extension pour prendre en compte les contraintes individuelles de chacune des exploitations est assez naturelle.

L'environnement de résolution du problème est constitué des différentes parcelles à occuper caractérisées par leur taille, pente et type de sol. Des indices de sensibilité au risque sont évalués pour chaque parcelle afin d'établir les connexions hydrologiques de voisinage et de comprendre la logique de propagation du risque entre les parcelles du bassin versant.

Un groupe d'agents est associé à chaque type de culture. A chaque groupe correspond un ensemble d'agents qui cherchent à réaliser son objectif de production en occupant les meilleures parcelles. Un paramètre de satisfaction caractérise chaque groupe et permet de chiffrer la distance qui le sépare de son objectif de production. Ce paramètre est pris en compte pour favoriser les groupes les plus éloignés de leur objectif. Chaque parcelle du territoire peut être occupée par un agent d'un des groupes et contribue ainsi à la réalisation de son objectif. En revanche, les agents s'interdisent à effectuer des permutations qui dégradent la valeur du risque dans le bassin versant. Une contrainte anti-érosive est également prise en compte pour interdire de placer sur les parcelles les plus sensibles du bassin versant une culture plus ruisselante que celle qui y existait auparavant. Ceci étant afin d'éviter de cultiver une culture ruisselante sur une parcelle très sensible puisque cette culture ne poussera pas dans ce cas [JAZIRI 02b]. La quantification du risque est évaluée à l'échelle du modèle numérique du terrain par l'automate cellulaire. Nous renvoyons à l'annexe 4 pour de plus amples détails sur le modèle multi-agents développé.

Le principe de l'algorithme d'optimisation est basé sur les étapes suivantes :

Algorithme de simulation :

Début

1. Sélection de la parcelle la plus sensible du territoire, non encore traitée.
2. Recherche de toutes les parcelles dont la permutation avec celle sélectionnée en 1 permet l'amélioration de la valeur du risque calculé au niveau du M.N.T.
3. Eliminer les parcelles dont les groupes occupants dépassent les seuils maximums de satisfaction autorisés en cas de permutation.
4. Classer les parcelles retenues en fonction de la satisfaction qu'elles procureraient aux groupes de leurs cultures occupantes si la permutation était effectuée.
5. Effectuer la permutation pour la meilleure des parcelles des étapes 2 à 4.
6. Aller en 1 tant qu'il existe une parcelle non encore traitée.

Fin

IV.2. Résultats de simulation

Le traitement d'une parcelle correspond à la recherche d'une permutation de sa culture avec celle d'une autre parcelle moins sensible du territoire, dans le but d'améliorer la valeur du risque. Une simulation consiste à traiter (une fois) l'ensemble des parcelles du territoire par la recherche de permutations possibles entre les cultures occupantes. Les résultats obtenus permettent de constater l'amélioration de la valeur du risque de ruissellement (figure 41) et la satisfaction des objectifs de production pour les différents types de culture (figure 42). Les surfaces en prairie (culture anti-ruisselante) ont augmenté au détriment des occupations très ruisselantes, notamment le maïs et le blé (figure 42) ce qui explique la diminution de la valeur du risque (figure 41). Notre système converge au bout de quelques simulations vers un état stable caractérisé par l'absence de permutations entre les agents.

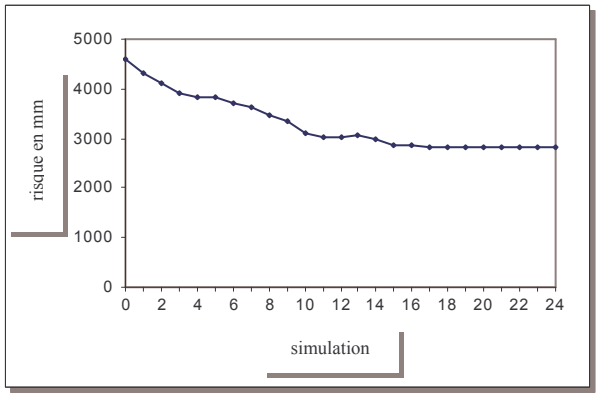


Figure 41 : Evolution de la valeur du risque au cours de 24 simulations.

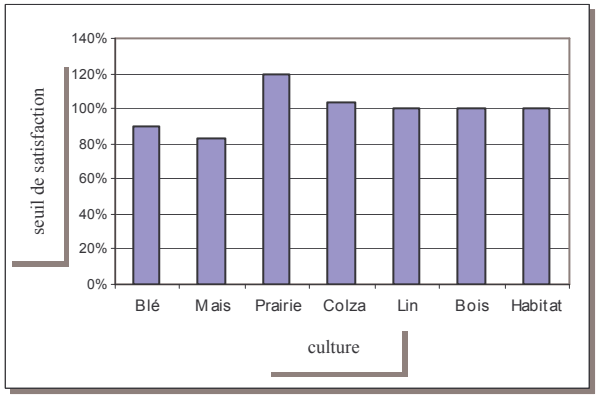


Figure 42 : Taux de satisfaction (en pourcentage par rapport à l'objectif) pour chaque type de culture.

Par ailleurs et pour pouvoir présenter nos résultats aux experts agronomes afin de mieux juger leur conformité aux exigences agronomiques, spatiales et hydrologiques, nous avons choisi de visualiser sous forme de cartes, la répartition des occupations de sol sur les parcelles avant et après le déroulement de notre système (figures 43, 44, 45 et 46)

Comme le montrent les figures 45 et 46 (zones encerclées), le système dissocie les regroupements de cultures ruisselantes afin de limiter le risque et affecte des cultures anti-ruisselantes dans les zones sensibles du territoire et dans le sens d'écoulement de l'eau (figure 44). Les cartes étaient présentées aux experts agronomes [GAILLARD 02] qui ont jugé le placement spatial de cultures cohérent et conforme aux exigences agronomiques et hydrologiques.

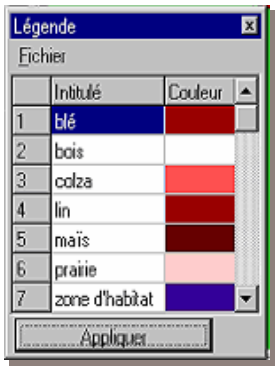


Figure 43 : Couleur attribuée pour chaque type de culture pour visualiser les cartes.

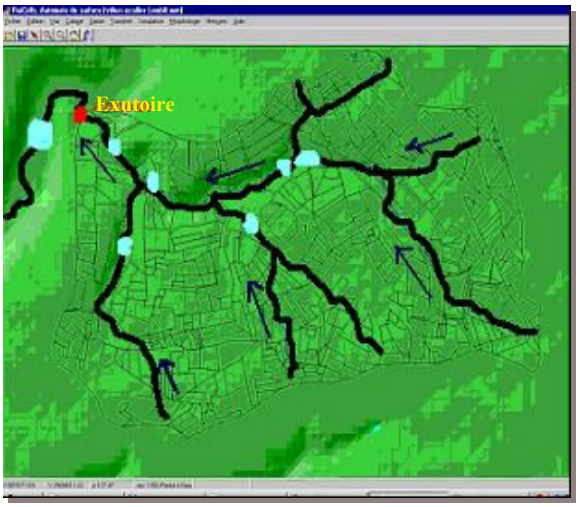


Figure 44 : Zones inondables (en rond) et sens du passage de l'eau à travers le bassin versant.

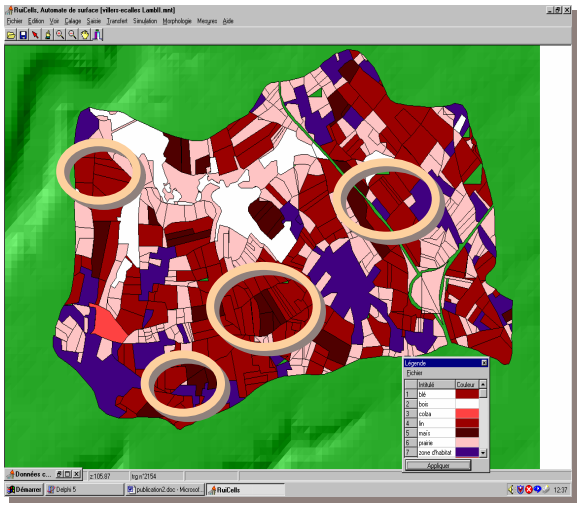


Figure 45 : Carte de répartition initiale des cultures sur les parcelles du territoire de Villers-Ecalles.

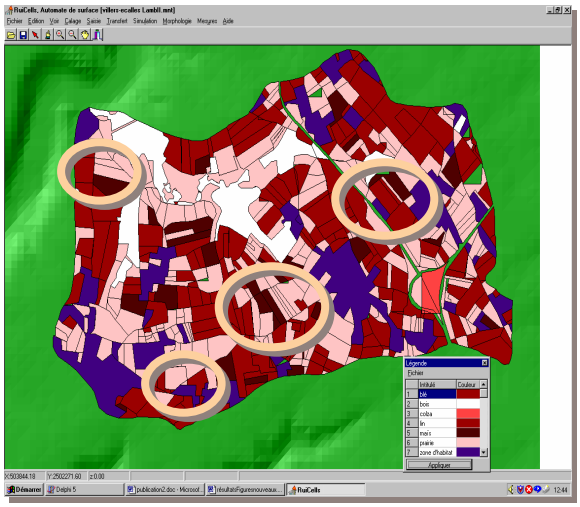


Figure 46 : Carte de répartition des cultures proposée par le système.

IV.3. Discussion

L'approche suivie pour la résolution du problème entre plutôt dans le cadre de l'optimisation classique (mono-critère). Le modèle développé n'est pas orienté satisfaction des contraintes puisque aucune analyse globale n'a été effectuée à ce niveau. Seul un ensemble restreint de contraintes, en l'occurrence les contraintes de production et les exigences agronomiques de pente, taille et type de sol, est pris en compte. La satisfaction de ces contraintes est assurée explicitement en procédant à une élimination successive des

affectations ne vérifiant pas les contraintes afin de limiter l'espace de recherche[81]. Le modèle simplifié du problème traité accorde via son aspect sous-contraint, une grande marge de manœuvre pour le système et explique l'amplitude des modifications apportées sur la répartition des cultures dans le territoire (figures 45 et 46). Cependant, cette démarche de résolution n'est pas tout à fait réaliste dans le sens où elle suppose une mise en commun de la ressource naturelle (les parcelles des exploitations) entre les différents agriculteurs et ignore un grand nombre de contraintes décideurs. Elle permet toutefois d'examiner ce qui serait possible de faire dans le meilleur des cas.

V. Seconde approche : Approche générique d'optimisation sous contraintes

Durant la première expérimentation, un certain nombre de contraintes ont été ignorées, en l'occurrence le cycle d'assolement des parcelles, les successions culturales et l'unité décisionnelle de la parcelle (l'ensemble des exploitations était supposé propriété d'un seul exploitant). Ces contraintes représentent en majorité des préférences dures pour les décideurs qui n'étaient pas intégrés dans le processus de décision. A cet effet, nous procédons à une démarche générique plus réaliste accordant une large place à la tâche de satisfaction des contraintes et à l'intégration des décideurs dans la définition et la négociation de leurs besoins. Nous étudions dans cette partie une deuxième expérimentation effectuée sur le bassin versant de Haute-Durdent (Pays de Caux). Nous nous basons sur les différents concepts développés tout au long de ce travail afin de mieux gérer les différentes contraintes applicatives et proposer ainsi des solutions plus réalistes et plus adaptées aux besoins et préférences des décideurs.

Le bassin versant d'étude, appelé bassin de Haute-Durdent, est situé dans la zone de Grainville-la-Teinturière comportant 1734 hectares dont 86,51% (1500 hectares) de surfaces cultivées et 13,49% (234 hectares) de villages. Nous disposons de 5 bassins versants élémentaires situés en amont du bassin versant de Grainville, d'une superficie totale d'environ 16 km² [DELAHAYE 95]. Cet exemple a été retenu pour la démonstration car il est représentatif de tous les cas de figures rencontrés et possède une morphologie d'ensemble caractéristique des situations existantes en Pays de Caux. Le bassin de Haute-Durdent est composé de 450 parcelles, dont 84 parcelles contenant une occupation fixe (bois ou village). L'ensemble du bassin est partagé entre 13 types d'occupations : betterave, blé, bois, colza,

[81] Nous retrouvons ici les concepts des techniques de filtrage (partie II.1 du chapitre 2).

culture fourragère, escourgeon, jachère, lin, maïs, pomme de terre, pois, prairie (sth) et zone d'habitat (village). Les zones boisées et d'habitat sont considérées fixes et ne sont donc pas prises en compte dans le processus de ré-affectation. La traduction de notre application en CSOP prend la forme suivante :

- $X = \{p_1, p_2, ... p_{450}\}$ ensemble des 450 parcelles du territoire ;
- $D = \{dom(p_i) \mid p_i \in X\}$ ensemble des 450 domaines associés aux parcelles (i.e., leurs valeurs possibles) ;
  - D_1 = {zone d'habitat, bois} ;
  - D_2 = {betterave, blé, colza, culture fourragère, escourgeon, lin, maïs, pomme de terre, pois, jachère, prairie} ;
- $C = \{c_1, ..., c_j, ..., c_{22}\}$ ensemble des 22 contraintes applicatives ci-dessus présentées ;
- F : Nous cherchons à améliorer la valeur du risque dans le bassin versant et la satisfaction des contraintes dures.

V.1. Mécanisme de résolution

Une solution à notre problème est une affectation des cultures sur les parcelles du territoire, qui minimise le risque et qui améliore au maximum la satisfaction des contraintes dures exigées par les décideurs. Le choix de la méthode doit assurer un certain compromis issu de la contradiction entre la complexité des contraintes exigées par les décideurs et l'obligation de respecter un temps de réponse raisonnable. Dans notre contexte de gestion de risque, la combinatoire étant énorme, une recherche exhaustive ou exacte ne peut être appliquée. A cet effet, nous avons élaboré une méthode hybride basée sur des principes inspirés de la stratégie évolutive [BÄCK 96] et du recuit simulé [KIRKPATRICK 83]. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche des méthodes de voisinage et d'évolution des algorithmes évolutifs.
Le processus de résolution consiste à effectuer un ensemble de simulations dans le but de proposer des solutions de ré-affectation assurant à la fois l'amélioration du risque et la satisfaction des contraintes dures.

V.1.1. Stratégie d'optimisation

Nous considérons la répartition réelle des cultures effectuée par les agriculteurs chacun sur ses propres parcelles, comme configuration de départ. Nous manipulons une seule

configuration (ou individu par analogie avec les stratégies évolutives) à la fois. Le processus de permutation de cultures entre deux parcelles joue le rôle d'opérateur de mutation, permettant ainsi de générer, à chaque itération, un individu enfant à partir de l'individu parent. A partir de la configuration initiale, le processus d'optimisation répète une procédure itérative d'exploration du voisinage inspirée de la méthode du recuit simulé (partie II.2.2 du chapitre 2). L'objectif consiste à chercher des configurations de coût plus faible en terme de risque tout en acceptant de manière contrôlée des configurations qui dégradent cette fonction de coût. La nouvelle configuration est acceptée systématiquement si la valeur du risque de ruissellement diminue. Sinon, l'acceptation d'une nouvelle configuration dont la valeur de coût est supérieure à celle de la configuration courante est déterminée de manière probabiliste : un réel $0 \leq \theta < 1$ est tiré aléatoirement et ensuite comparé avec une probabilité d'acceptation $p(\Delta E,T)$ fonction de la valeur de dégradation ΔE et de la température courante T. La température est contrôlée par une fonction décroissante qui définit un schéma de refroidissement. Le système part avec une température élevée qui tend vers 0 au fur et à mesure de l'avancement dans le processus de résolution afin d'emmener le système dans un état stable. Si $\theta \leq p(\Delta E,T)$, alors le nouvel état est accepté pour remplacer l'état courant, sinon, l'état courant est maintenu et réutilisé pour générer une nouvelle configuration. L'acceptation des dégradations permet de sortir des états stables de non-amélioration du risque et favorise l'exploration d'un espace de recherche plus vaste dans le but d'éviter une convergence prématurée vers un minimum local. La probabilité d'acceptation de dégradation étant faible afin d'éviter une dispersion aléatoire de la recherche de solutions.

V.1.2. Stratégie de satisfaction

Nous nous inspirons dans cette expérimentation de l'algorithme de simulation développé (partie II.2.3 du chapitre 4) tout en favorisant la satisfaction des contraintes dures. Afin d'éviter une dégradation des taux de satisfaction de ces contraintes, nous traitons (par permutation) seulement les parcelles dont l'occupation ne satisfait pas au moins une contrainte dure. Les parcelles satisfaisant toutes les contraintes dures restent invariantes (gardent la même occupation) au cours de la simulation afin de maintenir le taux initial de satisfaction. Ce choix de favoriser les contraintes dures s'explique par le fait que les mêmes parcelles peuvent être impliquées simultanément dans des contraintes dures, flexibles et secondaires. Le traitement de l'ensemble des parcelles violant des contraintes peut engendrer

l'amélioration de la satisfaction des contraintes flexibles et secondaires en dépit des contraintes dures, qui restent au point de vue des décideurs les plus exigeantes.
Nous attribuons une pénalité à chaque parcelle violant une contrainte dure. A partir de la configuration initiale, le mécanisme de parcours du voisinage est basé sur une permutation des occupations entre deux parcelles pénalisées. Le choix de la parcelle à permuter s'effectue suivant le nombre de pénalités afin de traiter en premier celles qui causent le plus de problèmes vis-à-vis de la satisfaction des contraintes dures.

Le principe de l'algorithme de résolution est le suivant :

Algorithme :

Début

- Partir de la répartition initiale $\mathbf{x_i}$ des cultures sur les parcelles.

`/* Phase d'initialisation */`

- Evaluer la satisfaction initiale des contraintes
- Extraire l'ensemble **P** des parcelles violant des contraintes dures /* Si toutes les contraintes dures sont satisfaites, extraire les instances violant des contraintes flexibles ou secondaires mais qui ne sont pas impliquées dans des contraintes dures afin de ne pas dégrader la satisfaction de ces dernières */

`/* Effectuer n simulations */`

- **Répéter**

  `/* Phase d'optimisation */`

  - Choisir une température initiale élevée **T** et une fonction de réduction de température α.
  - **Pour** chaque parcelle $\mathbf{p_i}$ appartenant à **P** ($\mathbf{p_i}$ étant la parcelle la plus pénalisée).
    - Générer un voisin **x** (par permutation de $\mathbf{p_i}$ avec une autre parcelle pénalisée).
    - Si le critère d'évaluation s'améliore alors $\mathbf{x_i} = \mathbf{x}$.
    - Sinon accepter la nouvelle configuration avec une probabilité aléatoire
    - Réduire la température $\mathbf{T = \alpha(T)}$

  `/* Phase de satisfaction */`

  - Mesurer les taux de satisfaction des contraintes
  - Mettre à jour l'ensemble **P** des parcelles pénalisées.
  - Retenir la solution si elle répond aux exigences fixées par le système.

- **Jusqu'à** condition d'arrêt.

Fin

A l'issue de chaque phase d'optimisation, le système détermine les taux de satisfaction des contraintes et le temps de réponse. Les taux de satisfaction sont déterminés à partir de l'interrogation de la base de données[82] via des requêtes SQL (Structured Query Language) déduites des différents composants des contraintes saisies par les décideurs (partie II.2.4 du chapitre 4). Si les taux de satisfaction sont supérieurs à un certain seuil (fixé par les décideurs ou par le système), la solution intermédiaire sera retenue et le système procède à une deuxième phase d'optimisation et ainsi de suite (tant qu'une condition d'arrêt n'est pas atteinte). L'ensemble des solutions est proposé aux décideurs qui choisiront celles qui correspondent le mieux à leurs attentes, suivant 4 critères : le temps de réponse, la valeur de risque, les taux de satisfaction et la répartition des cultures sur les parcelles.

V.1.3. Evaluation de la fonction de coût (valeur de risque)

L'utilisation de l'automate cellulaire dans la première expérimentation a montré sa grande utilité en tant qu'outil de simulation dynamique, permettant une estimation assez fine de la valeur du risque. Cependant, il va sans dire que cet outil de simulation du risque au niveau du M.N.T. est assez gourmand en temps de calcul[83]. En effet, étant donné la combinatoire du problème d'une part et le temps de simulation des phénomènes de ruissellement d'autre part, l'évaluation systématique de la valeur précise du risque à l'aide de l'automate devient très vite pénalisante dans la stratégie d'optimisation la plus rigoureuse. Le temps de simulation peut finalement être déterminant dans une perspective de développement d'outils interactifs d'aide à la décision pour l'optimisation sous contraintes.

Compte tenu de ces exigences, et afin de remédier à ce problème de temps de simulation, notre réflexion nous a conduit à définir un nouveau critère de risque qui permet une évaluation plus rapide de la valeur de ruissellement afin d'éviter les appels à l'automate gourmand en temps de calcul. Nous définissons une fonction de coût qui permet d'estimer la variation de ruissellement induite par une permutation des cultures sur deux parcelles quelconques. Ce critère approximatif de ruissellement, fonction des caractéristiques des parcelles, n'est qu'une estimation relativement grossière puisque son évaluation ne prend en compte que l'état de culture des parcelles et leur sensibilité au ruissellement et ignore le transfert de risque entre les zones du territoire.

[82] La base de données est implémentée en SGBD relationnel.
[83] Avec un processeur Pentium 4 et un M.N.T au pas de 75 m, une simulation du risque par l'automate cellulaire nécessite une dizaine de secondes sur un bassin versant de 15 km^2.

V.2. Résultats de simulation

Nous disposons de 27 contraintes décideurs dont 16 dures, 5 flexibles et 6 secondaires (figure 47). Ces contraintes sont saisies par les différents agriculteurs dans leur exploitation respective. La phase de négociation a permis de supprimer toutes les contraintes de complexité 'Surélevée' et celles de niveau de préférence 'Secondaires' (figure 48).

	Constante	Linéaire	Surélevée	Total
Dures	3	12	1	16
Flexibles	1	0	4	5
Secondaires	3	2	1	6
Total	7	14	6	27

Figure 47 : Liste des contraintes exigées par les différents décideurs.

	Constante	Linéaire	Surélevée	Total
Dures	3	12	0	15
Flexibles	1	0	0	1
Secondaires	0	0	0	0
Total	4	12	0	16

Figure 48 : Liste des contraintes retenues pour la satisfaction.

Différents tests ont été effectués afin de suivre l'évolution des critères alphanumériques de risque, de satisfaction des contraintes et du temps de réponse ainsi que le critère spatial de répartition des cultures sur les parcelles du territoire. La figure 49 montre une nette diminution de la valeur du risque de ruissellement qui, contrairement au temps de réponse, tend à diminuer au cours des simulations.

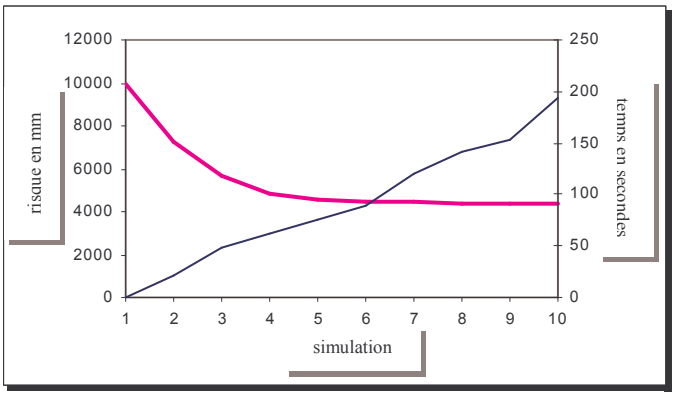


Figure 49 : Evolution du risque (en gras) et du temps de réponse au cours des simulations.

Ce résultat témoigne de l'efficacité de l'algorithme d'optimisation basé sur l'acceptation de quelques dégradations de la valeur du risque ce qui a permis de sortir des minimums locaux et d'obtenir des résultats tout à fait intéressants en un temps de simulation raisonnable

(quelques minutes pour une simulation). Ces dégradations sont acceptées au cours de la phase d'optimisation lors des permutations des cultures entre les parcelles pénalisées. Elles ne sont pas illustrées par la figure 49 qui visualise l'évolution de la valeur du risque au niveau des simulations et non au niveau des permutations.

V.2.1. Evaluation de la satisfaction des contraintes

Afin d'évaluer la satisfaction des contraintes prises en compte dans le processus de résolution, nous avons suivi l'évolution du taux moyen de satisfaction par rapport à toutes les contraintes dures et flexibles (figure 50). Les contraintes dans la première approche (partie IV) sont considérées satisfaites (à 100%) ou non satisfaites (à 0%), ce qui ne favorise pas l'analyse des causes de l'insatisfaction de ces contraintes et la détermination des variables non satisfaisantes. Une contrainte est satisfaite à 100% si toutes les variables qu'elles lient satisfont sa condition de satisfaction. Nous disposons dans notre approche de quelques centaines de variables pouvant être impliquées simultanément dans les mêmes contraintes. Une contrainte complexe impliquant un grand nombre de variables, ne peut souvent être satisfaite par toutes les variables impliquées (seul un sous-ensemble de variables liées satisfont la contrainte). Nous considérons dans ce cas qu'elle est satisfaite à un certain degré.
A cet effet, nous procédons à une évaluation en pourcentage de la satisfaction des contraintes (voir chapitre 4). Nous cherchons à suivre l'état de satisfaction des contraintes dures et à déterminer les variables à l'origine de leur insatisfaction pour mieux gérer (ré-affecter) ces variables et pouvoir orienter le processus d'optimisation pour assurer la tâche de satisfaction et apporter ainsi des solutions aux problèmes d'optimisation sous contraintes.

L'heuristique définie, qui consiste à ne permuter que les parcelles qui ne satisfont pas au moins une contrainte dure, a permis de maintenir (et même d'améliorer) le taux initial de satisfaction de ces contraintes. Cependant, nous remarquons quelques dégradations du taux de satisfaction notamment au niveau des simulations 3 et 5. En effet, à chaque simulation, le processus de résolution peut accepter une dégradation du taux de satisfaction pour ne pas tomber dans un maximum local[84]. Afin de mieux analyser ces chutes, nous procédons à l'évaluation des taux individuels de satisfaction de chaque contrainte dure pour suivre de près leur évolution au cours des simulations. La figure 51 illustre l'évolution des contraintes dures les plus significatives.

[84] Le système accepte des dégradations du taux de satisfaction par rapport à la dernière configuration mais n'autorise en aucun cas la descente au-delà du taux de satisfaction initial (avant la première simulation). Dans un tel cas, la solution générée est rejetée par le système et ne sera pas proposée aux décideurs.

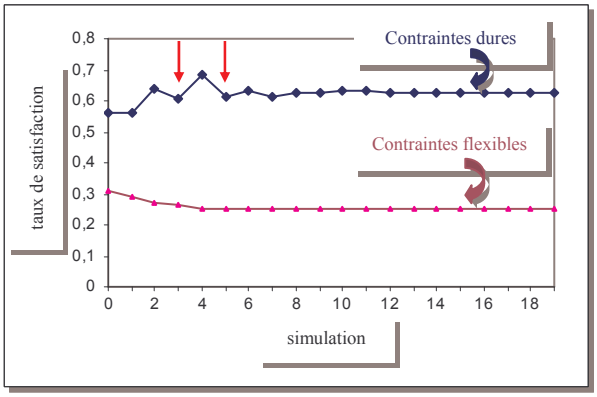


Figure 50 : Evolution du taux moyen de satisfaction des contraintes dures au cours des simulations.

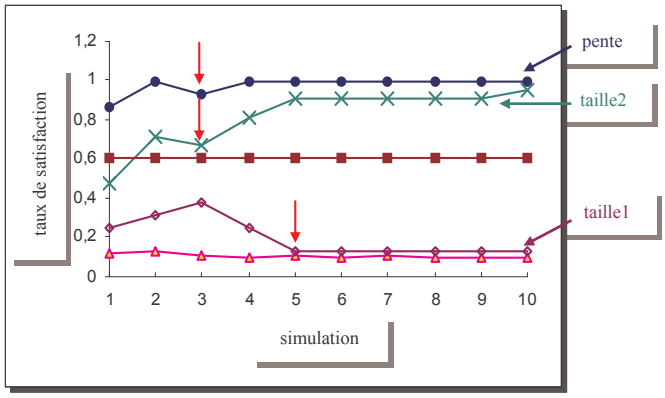


Figure 51 : Evolution du taux de satisfaction de quelques contraintes dures.

Nous constatons une chute du taux de satisfaction des contraintes 'pente' (instance de la contrainte applicative 1) et 'taille2' au niveau de la simulation 3 et de la contrainte 'taille1' lors de la simulation 5. 'taille1' et 'taille2' représentent deux instances de la contrainte applicative 3, exigées par deux décideurs différents. Ces chutes montrent que l'heuristique implémentée favorise l'amélioration de la satisfaction des contraintes dures en passant par la réduction du nombre de variables pénalisées, sans toutefois garantir l'amélioration de la satisfaction de toutes les contraintes dures (cf., contrainte taille1). Nous pensons que ces résultats sont assez logiques du fait que la même variable peut être impliquée dans plusieurs contraintes en même temps et que chaque contrainte implique généralement plusieurs variables simultanément. En effet, la ré-affectation d'une variable impliquée dans deux contraintes différentes, peut avoir un effet divergent sur leurs satisfactions respectives (entraîner l'amélioration de l'une et la dégradation de l'autre).

Par ailleurs, et contrairement aux contraintes dures, la satisfaction des contraintes flexibles ne s'améliore pas et tend même à se dégrader au cours des simulations (figure 51). Ce résultat attendu est une conséquence logique de la démarche de résolution orientée pour l'amélioration de la satisfaction des contraintes dures. En effet, l'algorithme extrait seulement les parcelles qui violent des contraintes dures et qui représentent l'entrée de la phase d'optimisation. Ces parcelles peuvent également être impliquées dans des contraintes flexibles ou secondaires et leur traitement peut influencer la satisfaction de ces contraintes. Cette démarche peut être rectifiée pour traiter les parcelles qui violent des contraintes flexibles ou secondaires afin de maintenir stable la satisfaction des contraintes dures. Cependant, dans notre problème mixte d'optimisation et de satisfaction, nous cherchons simultanément à optimiser le risque et à satisfaire les contraintes dures ce qui nous a poussé à

privilégier la première approche. Toutefois, nous faisons recours à la deuxième approche pour traiter les parcelles qui violent des contraintes flexibles ou secondaires quand toutes les contraintes dures sont initialement satisfaites. Ces parcelles ne doivent pas être impliquées dans des contraintes dures afin de ne pas détériorer la satisfaction de ces dernières.

Une autre alternative de résolution consiste à combiner ces deux démarches pour traiter en deux phases les parcelles violant l'ensemble des contraintes dures, flexibles et secondaires. Il s'agit de traiter dans une première phase de simulation, les parcelles violant des contraintes dures. Après que le système ait convergé et qu'il n'y ait plus de permutations possibles entre ces parcelles, le système procède à une deuxième phase de simulation pour traiter les parcelles violant des contraintes flexibles ou secondaires et qui ne sont surtout pas impliquées dans des contraintes dures afin de ne pas influencer leur satisfaction.

V.2.2. Cohérence des résultats

Afin de vérifier la cohérence des résultats obtenus avec le comportement interne du système, nous procédons au suivi des modifications apportées sur les occupations des parcelles au cours du processus d'optimisation. Les tests effectués consistent à reprendre le déroulement des simulations et à évaluer le nombre de permutations effectuées au cours de l'optimisation et leur conséquence sur le nombre de parcelles pénalisées (figure 52). Le nombre de permutations de parcelles effectué au cours du même test tend vers 0 ce qui explique la convergence de notre système au bout de quelques simulations vers un état stable caractérisé par l'absence de permutations possibles entre les parcelles pénalisées. Le nombre de parcelles pénalisées diminue de 19,42% (de 139 à 112) au cours de 10 simulations (figure 53).

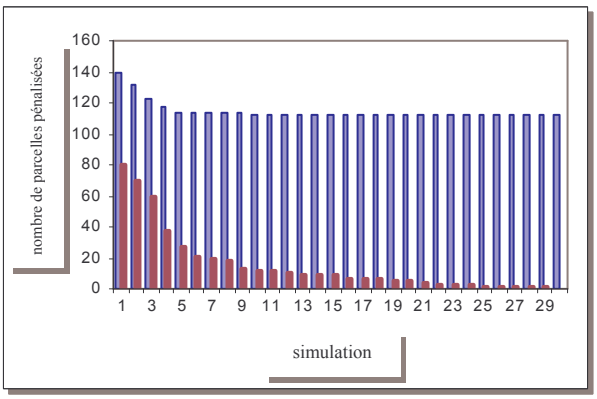


Figure 52 : Suivi du nombre de permutations (en rouge) et de parcelles pénalisées (en bleu) au cours des simulations.

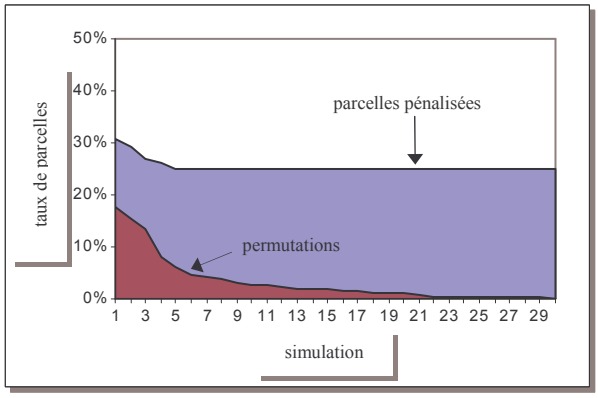


Figure 53 : Pourcentage de permutations et de parcelles pénalisées par rapport au nombre total de parcelles (450) dans le territoire.

Les résultats obtenus prouvent que les modifications apportées par le processus d'optimisation orientent certaines parcelles pénalisées vers la satisfaction des contraintes dures qui l'impliquent. Ceci témoigne de la pertinence de l'approche de résolution suivie et son apport pour l'amélioration simultanée de la valeur du risque et de la satisfaction des contraintes dures.

Le taux prometteur mais relativement faible de réduction du nombre de parcelles pénalisées peut être expliqué par la multitude de contraintes prises en compte, qui ont visiblement restreint la marge de manœuvre du système. Malgré ce nombre important de contraintes exigées par les décideurs et le nombre restreint de parcelles traitées (seulement celles violant des contraintes dures), la valeur du risque a diminué de 56% au bout de 9 simulations (de 9967 à 4377 mm, figure 49). Cette amélioration obtenue par le traitement de 30% du nombre total de parcelles du territoire (139 parmi 450, figure 53) prouve que l'aggravation du risque est due essentiellement à des parcelles élémentaires dont la mauvaise gestion par des agriculteurs individuels a modulé la gravité du processus pour l'ensemble de la collectivité. Nous pensons ainsi qu'il serait intéressant de suivre de près l'effet de la permutation de certaines parcelles individuelles sur la valeur du risque pour mieux confirmer cette hypothèse. La figure 54 montre quelques améliorations importantes de la valeur du risque observées au cours de la phase d'optimisation.

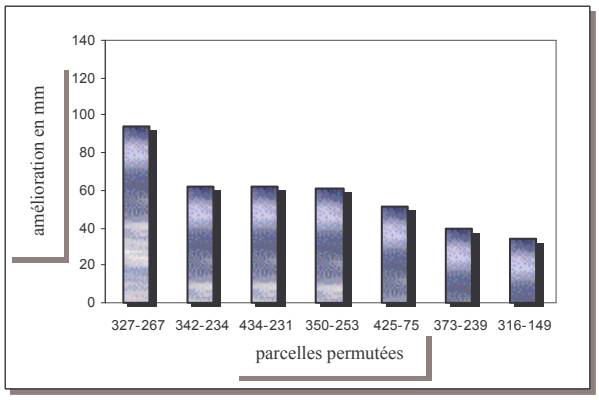


Figure 54 : Amélioration du risque à l'issue de quelques permutations de parcelles.

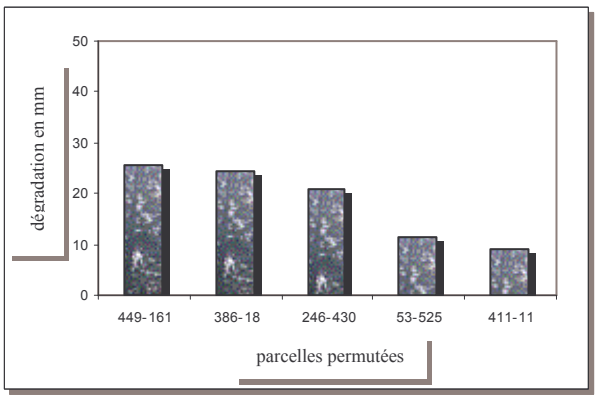


Figure 55 : Dégradation du risque à l'issue de quelques permutations de parcelles.

Cependant, les permutations des cultures effectuées entre les parcelles pénalisées n'améliorent pas forcément la valeur du risque. En effet, l'algorithme d'optimisation s'autorise à effectuer des permutations qui augmentent la valeur du risque afin d'explorer d'autres parties dans l'espace de recherche. Ces dégradations sont acceptées au cours de la

phase d'optimisation lors des permutations des cultures entre les parcelles pénalisées. La figure 55 illustre quelques dégradations de la valeur du risque suite à des permutations de parcelles au cours de la phase d'optimisation.

Par ailleurs, la courbe de la figure 49 comporte des états stables (au niveau des simulations 5 et 6) relatifs à la non amélioration de la valeur de ruissellement et montre que malgré les permutations effectuées par le processus d'optimisation (figure 52), le risque reste invariant. Certaines permutations de parcelles n'ont donc aucun effet sur la valeur du risque mais contribuent malgré cela à l'évolution du taux de satisfaction des contraintes (figure 51). Cette divergence entre l'évolution des critères de risque et de satisfaction des contraintes prouve l'aspect multicritère de notre problème et la complexité à laquelle nous étions confrontés lors de la conception de notre système d'optimisation sous contraintes.

V.2.3. Comportement du système vis-à-vis des contraintes

Nous avons constaté lors des tests précédents que le taux de satisfaction des contraintes dures tend à s'améliorer au cours du temps tandis que celui des contraintes flexibles (figure 50) varie légèrement[85] (par rapport à la variation de la satisfaction des contraintes dures). Cette variation peut être expliquée par la présence de parcelles violant des contraintes dures et qui sont également impliquées dans des contraintes flexibles. Le traitement de ces parcelles au cours du processus d'optimisation réaffecte leurs variables d'optimisations (les occupations) et influe sur la satisfaction des contraintes flexibles impliquant ces parcelles. Afin de confirmer cette pensée et mieux analyser le comportement des contraintes dures et flexibles au cours de la simulation, nous procédons à deux tests complémentaires. Dans le premier test, nous introduisons seulement trois contraintes dures pour suivre l'évolution de leur satisfaction au cours des simulations. Dans le deuxième test, nous introduisons ces mêmes contraintes mais en tant que flexibles afin de comparer leur satisfaction avec celle du premier test et analyser ainsi le comportement du système de simulation face aux mêmes contraintes mais avec des préférences différentes (figure 56). Notons que nous introduisons également des contraintes dures pour le deuxième test afin de traiter les parcelles violant ces contraintes au cours de la phase d'optimisation.

[85] En général vers la diminution.

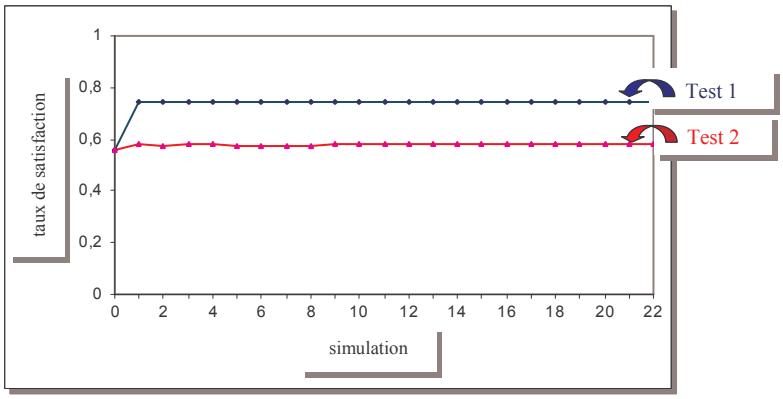


Figure 56 : Taux de satisfaction des 3 contraintes en tant que NP1 (Test1) et NP2 (Test2).

La figure 56 montre que le système de simulation favorise toujours l'amélioration de la satisfaction des contraintes dures quelles que soient ces contraintes. Le critère « préférence sur les contraintes » (défini au chapitre 3) oriente donc la résolution pour favoriser les contraintes les plus préférées des décideurs.

V.2.4. Pertinence de l'approche proposée

Afin de s'assurer de la robustesse de notre système et de son adaptation vis-à-vis de différentes combinaisons de contraintes, nous effectuons des expérimentations sur différents jeux de contraintes introduits par l'ensemble des décideurs (figures 57 et 58). Les tests 1, 2 d'une part et 3, 4 d'autre part diffèrent par quelques contraintes seulement (la plupart des contraintes prises en compte lors de ces tests sont les mêmes).

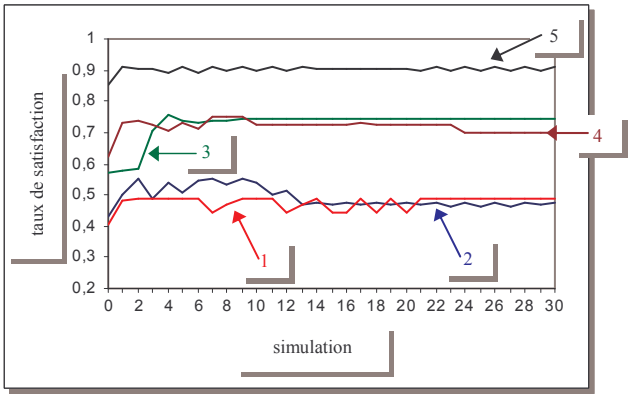


Figure 57 : Taux de satisfaction des contraintes dures au cours de 5 tests.

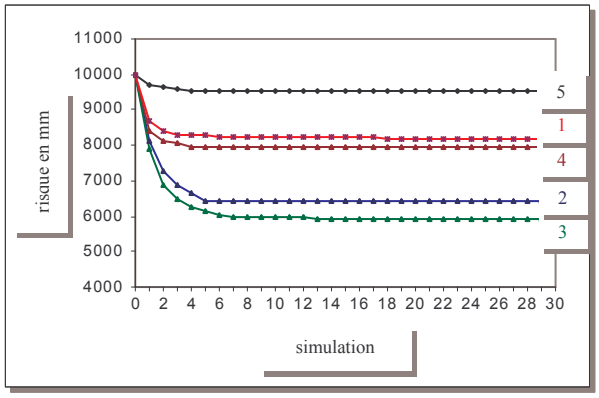


Figure 58 : Valeur du risque au cours des 5 tests.

Les résultats obtenus montrent une nette amélioration du taux moyen de satisfaction des contraintes dures pour l'ensemble des tests (figure 57). La valeur du risque diminue au cours

des simulations et converge vers un état stable (figure 58). Ainsi, le système de simulation assure simultanément les tâches d'optimisation et de satisfaction.

En effet, l'amélioration du risque est d'autant plus importante que la marge de manœuvre du système est grande. Un grand nombre de contraintes dures violées entraîne naturellement un nombre important de parcelles pénalisées, ce qui offre une certaine marge au système pour chercher des permutations de parcelles améliorant le risque. De ce fait, la légère amélioration de la valeur du risque au cours du test 5 (figure 58) peut être expliquée par le nombre restreint de parcelles pénalisées (figure 60).

Par ailleurs, nous remarquons la présence d'oscillations, notamment au niveau des tests 2 et 5 (figure 57). Les figures 59 et 60 montrent l'évolution du nombre de permutations et de parcelles pénalisées au niveau des tests 2 et 5.

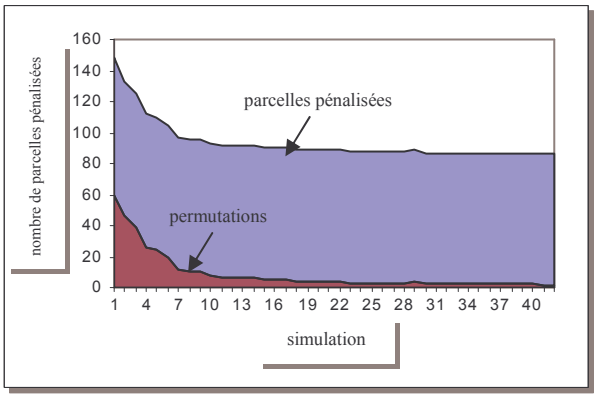


Figure 59 : Nombre de permutations de parcelles parmi celles pénalisées pour le test 2.

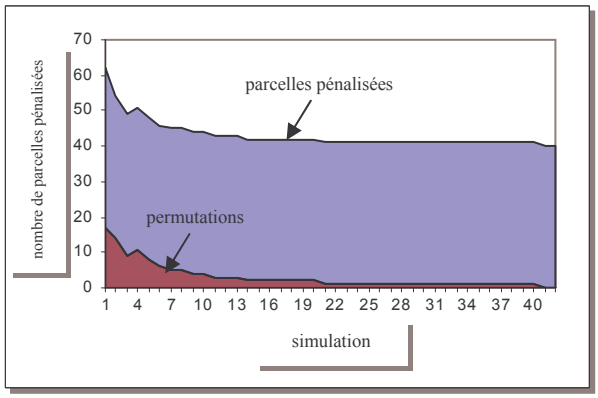


Figure 60 : Nombre de permutations de parcelles parmi celles pénalisées pour le test 5.

Les oscillations observées peuvent avoir deux origines :

- Le critère approximatif utilisé pour évaluer la valeur du risque au niveau du parcellaire (partie V.1.4) ne reflète pas l'état réel du risque de ruissellement dans le territoire. Ce critère imprécis, utilisé pour guider la phase d'optimisation au niveau de chaque simulation, cumule des imprécisions au cours de la résolution et fait diverger le système entre deux simulations successives d'où les oscillations.

- Le système génère des cycles que ce soit au niveau des permutations effectuées (la même parcelle peut retrouver son occupation initiale après un certain nombre de permutations) ou au niveau des solutions générées ($s \rightarrow s' \rightarrow s \rightarrow s'$). Ce problème est assez répandu dans la plupart des problèmes d'optimisation et a été traité par divers travaux [GLOVER 97]. Nous étions confrontés à ce même problème lors de la première expérimentation (partie IV. Modèle

expérimental d'optimisation). Pour empêcher de telles situations, nous nous sommes inspirés de la méthode Tabou [GLOVER 86] afin de mémoriser les dernières configurations explorées, et d'interdire les mouvements susceptibles de conduire à l'une de ces configurations.

V.2.5. Prise en compte de la satisfaction individuelle pour chaque décideur

Les contraintes prises en compte tout au long des tests effectués sont exigées par les différents décideurs impliqués dans le processus de décision, à savoir les agriculteurs, les agronomes et les instances locales. Nous nous somme toutefois limités à la visualisation du taux moyen de satisfaction des contraintes pour l'ensemble des décideurs (simultanément). Nous cherchons dans cette partie à suivre la satisfaction de contraintes pour chaque décideur, indépendamment des autres. Nous considérons trois décideurs agriculteurs qui saisissent chacun 4 contraintes dures relatives aux exigences de pente, de taille, de type de sol et des objectifs de production dans leurs exploitations respectives.

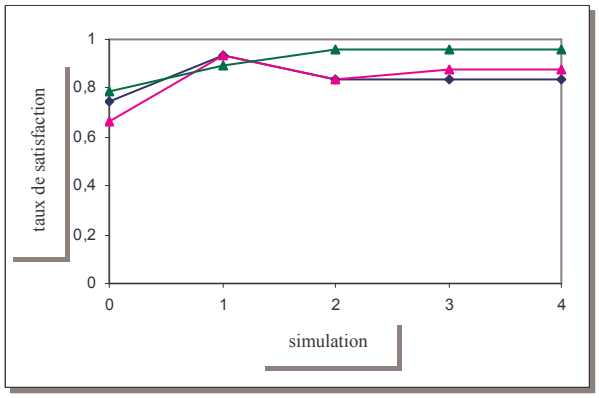


Figure 61 : Taux moyen de satisfaction des contraintes dures pour trois décideurs.

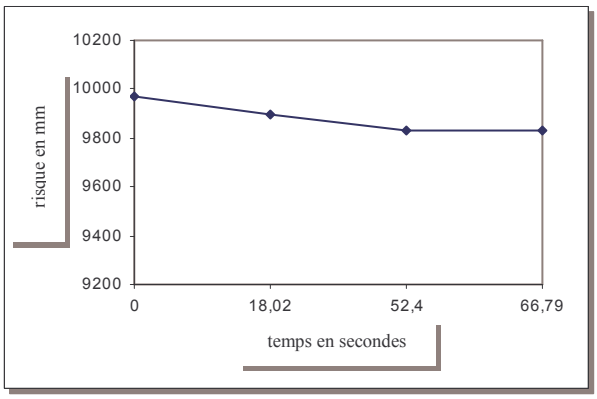


Figure 62 : Valeur du risque en fonction du temps de réponse pour l'ensemble du bassin versant.

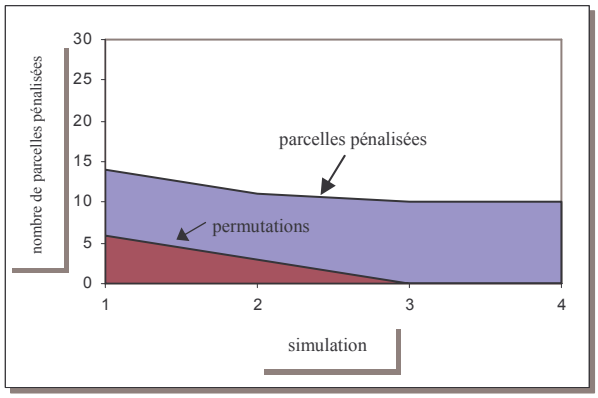


Figure 63 : Nombre de permutations de parcelles parmi celles pénalisées.

Les courbes obtenues prouvent que le système améliore les taux de satisfaction des contraintes pour l'ensemble des décideurs (figure 61). La convergence rapide du système (au bout de 4 simulations) ainsi que la faible diminution de la valeur du risque (de 9967 à 9828, figure 62) s'explique par le nombre limité de parcelles pénalisées (14 parmi 450) et prises en compte pour la permutation au cours du processus d'optimisation (figure 63).

V.2.6. Conformité agronomique

En revanche, pour s'assurer de la conformité des résultats vis-à-vis des exigences agronomiques et spatiales, nous avons visualisé sous forme de cartes, la répartition des occupations du sol sur les parcelles avant et après le déroulement du système (figures 64, 65, 66 et 67).

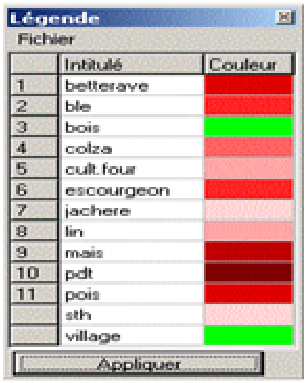


Figure 64 : La couleur attribuée pour chaque type de culture est d'autant plus foncée que la culture est ruisselante. Les occupations fixes sont représentées en vert.

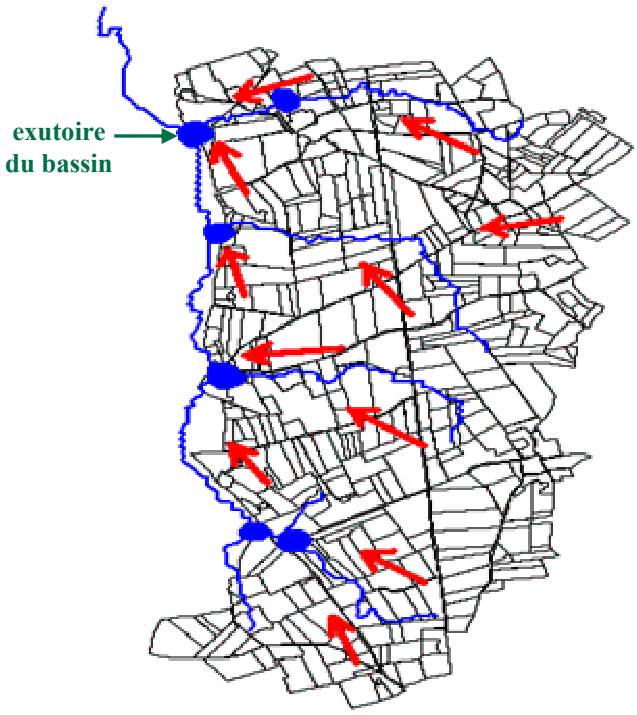


Figure 65 : Courbes d'écoulement, zones inondables (en bleu) et sens du passage de l'eau à travers le bassin versant.

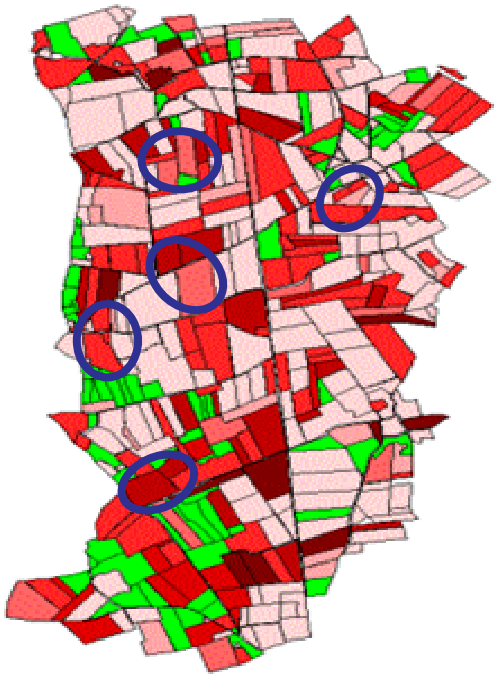


Figure 66 : Répartition initiale des cultures sur le bassin versant.

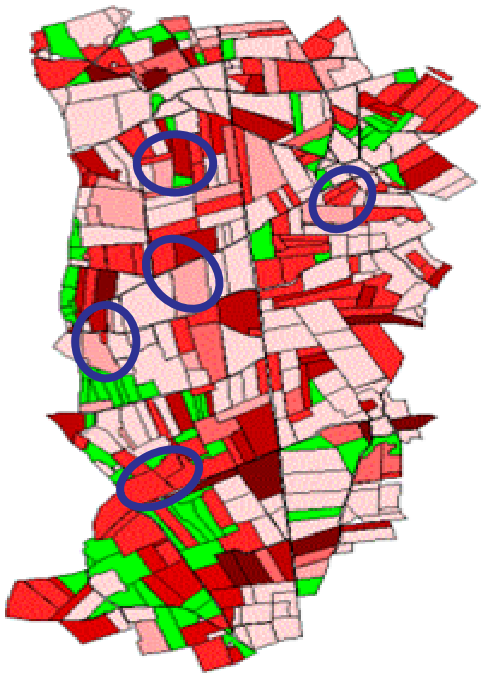


Figure 67 : Répartition des cultures proposée par le système après une simulation.

Le processus de résolution affecte des cultures anti-ruisselantes dans les zones sensibles du territoire et dans le sens d'écoulement de l'eau (figures 66 et 67, zones encerclées). Cependant, le nombre important de contraintes exigées par les décideurs et prises en compte dans le processus de satisfaction ainsi que l'utilisation d'une heuristique limitant le processus d'affectations au traitement des parcelles qui violent des contraintes dures (représentant 30% de l'ensemble des parcelles) explique le nombre limité, mais efficace, de ré-affectations apportées à la répartition des cultures. Le nombre restreint de contraintes dans la première expérimentation[86] ainsi que le traitement de toutes les parcelles du territoire (100% de 529 parcelles) ont permis une plus grande marge de manœuvre au système pour effectuer des modifications plus importantes sur la carte d'assolements (figures 45 et 46).

V.2.7. Solutions multicritères proposées

Etant dans un cadre d'aide à la décision, le système de simulation propose différentes solutions aux décideurs (figures 68, 69, 70 et 71), comportant chacune la valeur courante du risque, l'évaluation du taux moyen de satisfaction des contraintes, le temps de réponse et la répartition des cultures sur les parcelles du territoire. Nous présentons dans cette partie

[86] Notamment la possibilité de permuter des parcelles appartenant à des exploitations différentes.

quelques exemples de solutions générées (refusées ou acceptées) par le système de simulation.

Notons que le processus de résolution accepte quelques dégradations du taux de satisfaction pour ne pas tomber dans un maximum local. Cependant, les solutions générées, comportant un taux de satisfaction des contraintes dures inférieur au taux initial (avant le déroulement du système de simulation) sont rejetées par le système et ne seront donc pas proposées aux décideurs (cf. solution 1, figure 69).

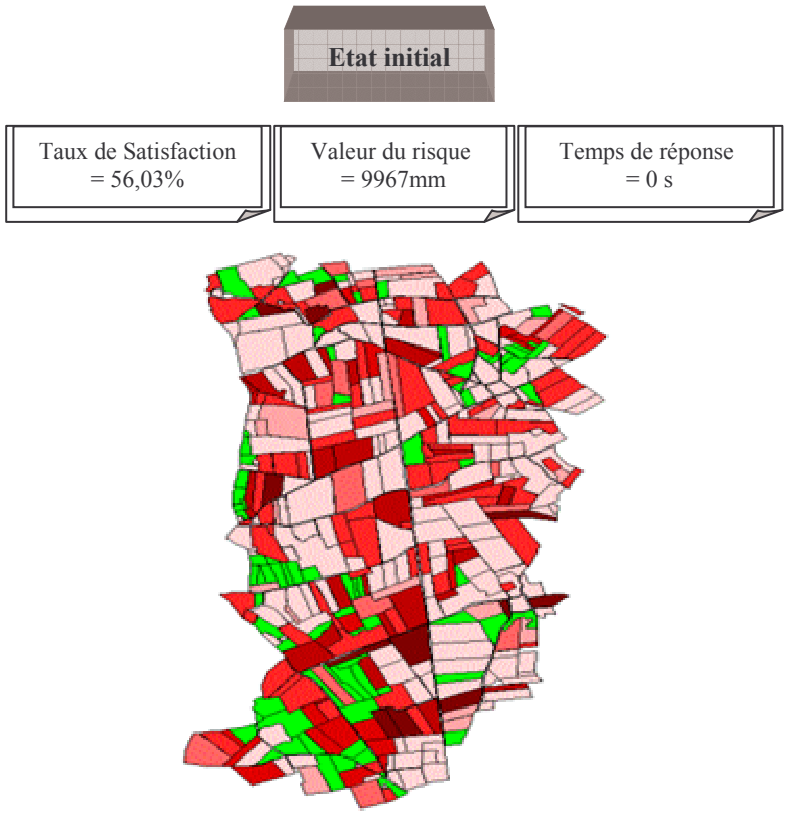


Figure 68 : Configuration initiale avant le déroulement du système.

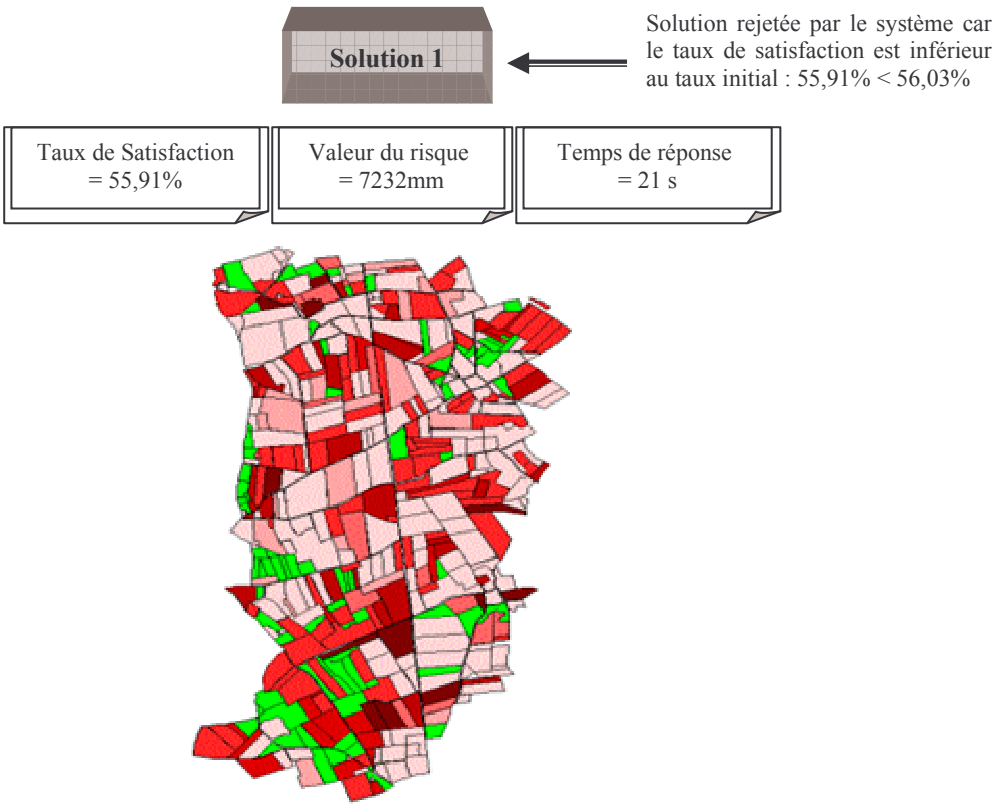


Figure 69 : Solution générée mais refusée par le système.

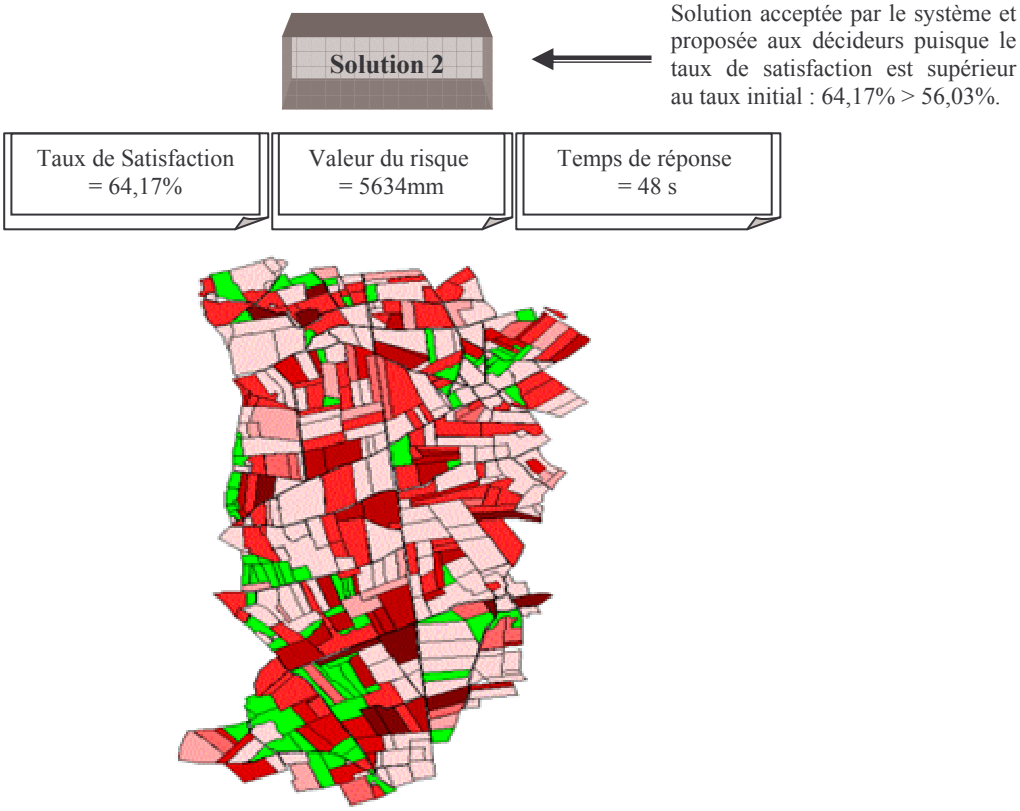


Figure 70 : Solution générée par le système et proposée aux décideurs.

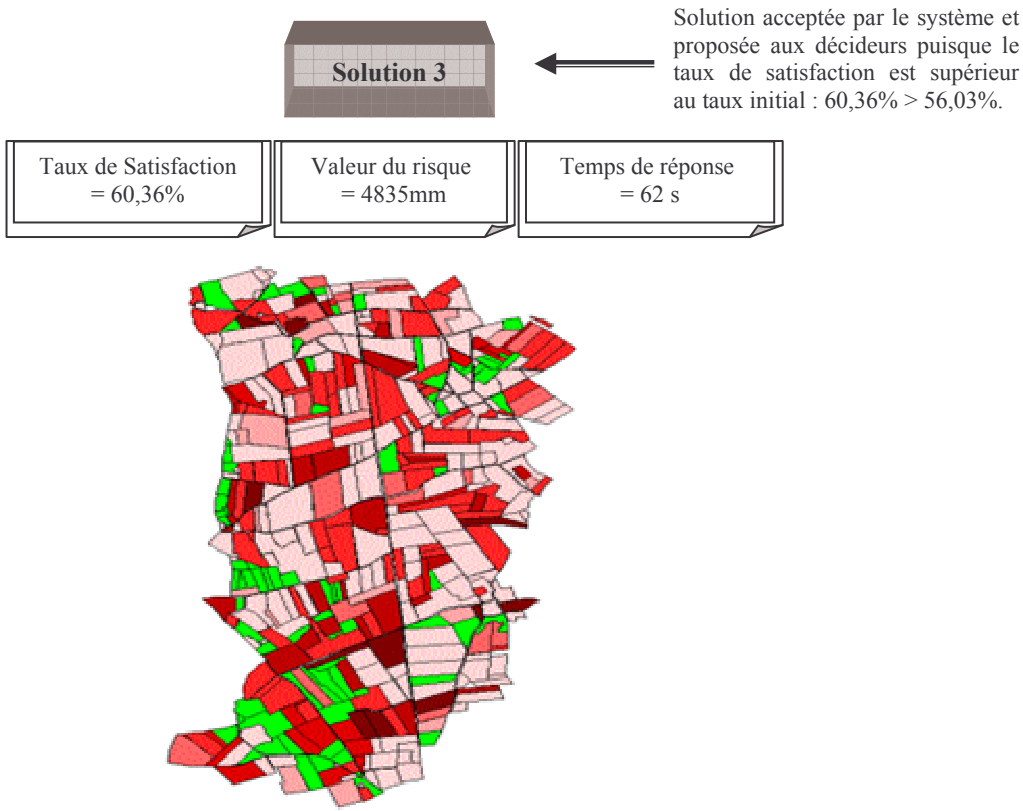


Figure 71 : Solution générée par le système et proposée aux décideurs.

VI. Synthèse

Nous avons mené dans ce chapitre une réflexion autour des problèmes de gestion de risque de ruissellement touchant les territoires agricoles. Deux expérimentations ont été effectuées sur les bassins versants de Villers-Ecalles et de Grainville la Teinturière, situés en Seine Maritime (France). Les résultats pratiques montrent clairement l'apport de la modélisation et de l'approche générique dans la génération de solutions appropriées et mieux adaptées aux besoins des décideurs. Cependant, bien que les objectifs que nous avons fixés soient atteints, à savoir la diminution du risque et l'amélioration de la satisfaction des contraintes dures, des expérimentations plus poussées doivent être poursuivies afin d'apporter une évaluation plus fine du système développé et de son efficacité face aux problèmes d'optimisation sous contraintes. Il serait également intéressant d'approfondir notre réflexion autour des problèmes d'optimisation sous contraintes afin d'améliorer le système de simulation développé et de l'adapter à des problèmes autres que l'aménagement du territoire.

CONCLUSION GENERALE :

Apport de notre approche et perspectives

Apport de notre travail

Le manque de définition et de modélisation des contraintes et d'analyse générique de leur complexité et de leur comportement au cours d'un processus de simulation constitue un point faible des travaux traitant des contraintes. Cet aspect fondamental et déterminant dans tout processus de décision et qui reste jusqu'à lors ignoré et mal défini, justifie cette réflexion pour développer des outils d'analyse et de modélisation génériques et interactifs sortant des approches classiques et spécifiques à un traitement particulier. Notre intérêt s'est porté essentiellement sur la proposition d'une approche générique de progression dans la résolution des problèmes complexes et sur-contraints. Notre approche modélise à la fois le processus d'acquisition, de formalisation, de négociation et de satisfaction des contraintes dans le cadre d'un processus d'optimisation.

L'approche proposée est développée dans un cadre mixte, tirant profit de trois disciplines aussi complémentaires que concurrentes telles que l'optimisation, la satisfaction des contraintes et l'aide à la décision. Cet aspect triparti relevant d'un couplage de techniques d'optimisation classique et d'analyse multicritère nous a permis de développer un système interactif pour la résolution des problèmes d'optimisation sous contraintes. Notre approche part des décideurs qui expriment leurs objectifs et contraintes, négocie avec eux ces contraintes et revient aux décideurs pour leur proposer les solutions établies en fonction de leurs choix et exigences.

L'aspect multidisciplinaire de notre travail, alliant des informaticiens, des géographes et des agronomes, nous a permis de mieux dégager les difficultés et les traitements complexes nécessaires à la prise de décision dans un environnement hétérogène. Cette collaboration entre des chercheurs de disciplines variées nous a été de grande utilité dans l'analyse et la spécification des divergences qui peuvent survenir lors de la communication et l'échange d'informations entre des acteurs de domaines de compétences et de pouvoirs d'expression différents.

D'un point de vue théorique, notre travail s'articule autour de quatre axes principaux, à savoir l'optimisation, la satisfaction de contraintes, l'aide à la décision et l'optimisation sous contraintes.

Notre contribution concernant l'aspect optimisation se situe au niveau de la définition d'un vocabulaire d'échange d'informations permettant de positionner le problème dans son cadre naturel mono ou multicritère et de déterminer ainsi les méthodes appropriées de résolution.

Notre réflexion concernant l'aspect satisfaction des contraintes nous a conduit à proposer une approche générique pour la classification et la formulation des contraintes en passant par une analyse de leur complexité et de leur évolution au cours d'un processus de simulation.
Notre apport concernant l'aspect interactif de notre modèle se situe au niveau de la démarche adoptée qui s'articule autour des décideurs. L'intégration de ces derniers dans la prise de décision dépasse la simple intervention au début du processus pour saisir leurs préférences sur les contraintes ou à la fin de la résolution pour évaluer les solutions proposées. Les décideurs interviennent tout au long du processus de modélisation et de résolution pour saisir leurs besoins et leurs préférences sur ces besoins, pour négocier leurs contraintes avec le système et enfin choisir la solution la mieux adaptée à leurs choix et préférences. Des opérations de manipulation des contraintes ont été définies afin de donner le moyen aux décideurs pour relaxer et mettre à jour leurs contraintes.
L'aspect optimisation sous contraintes a été traité via la formulation et le suivi de l'évolution de la satisfaction des contraintes dans le cadre d'un processus d'optimisation.

Dans le cadre pratique, nous avons mené une réflexion autour des problèmes de risque de ruissellement touchant les territoires agricoles afin d'apporter des solutions efficaces aidant les décideurs à mieux organiser leurs espaces individuels pour une meilleure gestion de leurs problèmes collectifs de risque. Les différentes expérimentations menées ont montré l'intérêt de l'approche générique proposée dans la génération de solutions satisfaisantes et mieux adaptées aux besoins des décideurs. Les résultats pratiques témoignent clairement de l'apport du système développé face aux applications complexes, tant sur la réalisation des objectifs visés que sur la satisfaction des contraintes décideurs et l'intégration de ces derniers dans la prise de décision.
Le choix du domaine de l'aménagement de territoire comme champ d'investigation peut être expliqué par l'ampleur et la diversité des applications assez variées ainsi que par l'environnement évolutif et complexe caractérisé par la multitude de facteurs sociaux, agronomiques et économiques influençant la prise de décision. Outre ces facteurs, la multitude d'acteurs de niveaux de perception différents ainsi que la diversité des contraintes qu'ils exigent et la complexité des traitements nécessaires à leur satisfaction, font que ce type de problèmes rencontrés sont en général sur-contraints.

Perspectives

Le travail présenté dans cette thèse propose un cadre générique interactif de modélisation des contraintes et des objectifs des décideurs pour la formulation et la résolution des problèmes d'optimisation sous contraintes. Le travail effectué durant ces trois années nous a permis de dégager les difficultés théoriques et pratiques rencontrées lors de la confrontation des applications complexes et multidisciplinaires. Les fondements génériques développés pour l'acquisition et la gestion des contraintes ont constitué un support formel pour la modélisation des différents aspects de la problématique traitée et la prise en compte de la divergence sémantique entre le monde réel et de simulation.

Cependant, cette première réflexion nécessite certaines améliorations, notamment pour prendre en compte d'autres formes de contraintes, en l'occurrence les méta-contraintes, dont la représentation est plus complexe et nécessite le recours à des traitements plus spécifiques. Nous envisageons également mener des expérimentations plus poussées afin de tester le système développé sur d'autres problèmes d'optimisation sous contraintes.
Par ailleurs, l'intégration des décideurs dans la prise de décision est assurée via une interaction analyste-décideur basée sur un processus de négociation des contraintes. L'interaction décideur-décideur n'est pas prise en compte directement dans le processus de simulation (qui se limite à la prévision des réactions des différents décideurs) et elle est gérée de façon centralisée en passant par le système. Il serait intéressant d'explorer cette voie et d'assurer ainsi une concertation directe entre les décideurs eux-mêmes.

BIBLIOGRAPHIE

[AARTS 97] E. H. L. AARTS et J. K. LENSTRA: "Local search in combinatorial optimization", Wiley, Chichester, 1997.

[AHUJA 02] R.K. AHUJA, Ö. ERGUN, J. B. ORLIN et A.P. PUNNEN: "A survey of very large-scale neighbourhood search techniques", Discrete Applied Mathematics, pages 75-102, 2002.

[ALQUIER 90] A.M. ALQUIER et V. PRINCE: "Cognitive modelling and design for knowledge dynamics in automatized information systems", Proceedings of Cognitiva-90, pages 655-661, Espagne, 1990.

[ARROW 86] J. K. ARROW et H. RAYNAUD: "Social choice and multi-criterion decision making", MIT press, Cambridge, 1986.

[AUZET 87] A.V. AUZET: "L'érosion des sols cultivés en France sous l'action du ruissellement", Annales de géographie, pages 529-556, 1987.

[BACCHUS 98] F. BACCHUS et P. VAN BEEK: "On the conversion between non-binary and binary constraint satisfaction problems", Proceedings of the 15th National Conference on Artificial Intelligence, American Association for Artificial Intelligence (AAAI'98), pages 326-333, Madison, USA, 1998.

[BÄCK 91] T. BÄCK, F. HOFFMEISTER et H-P. SCHWEFEL: "A survey of evolutionary strategies", Proceedings of 4th International Conference on Genetic Algorithms (ICGA'91), pages 2-9, USA, 1991.

[BÄCK 93] T. BÄCK et H.P. SCHWEFEL: "An overview of evolutionary algorithms for parameter optimization", Evolutionay Computation, pages 1-23, 1993.

[BÄCK 96] T. BÄCK: "Evolutionary algorithms in theory and practice", Oxford University Press, USA, 1996.

[BARNIER 02] N. BARNIER: "Application de la programmation par contraintes à des problèmes de gestion du trafic aérien", Thèse de doctorat, Institut National Polytechnique de Toulouse, France, 2002.

[BARTHES 98] B. BARTHES, G. DE NONI, E. ROOSE, J. ASSELINE, A. ALBRECHT et M. VIENNOT: "Influences du travail du sol et des apports sur le ruissellement et l'érosion : Le cas des Rougiers de Camarès dans le Sud-Aveyron", Orstom actualités, 1998.

[BATESON 77] G. BATESON: "Vers une écologie de l'esprit", Tome 1, Seuil (édition française), 1977.

[BELL 88] D. E. BELL, H. RAIFFA et A. TVERSKY: "Decision making: descriptive, normative, and prescriptive interactions", Cambridge University Press, Grande Bretagne, 1988.

[BELLMAN 57] R. BELLMAN: "Dynamic programming", Princeton University Press, USA, 1957.

[BELTON 90] V. BELTON: "Multiple criteria decision analysis practically the only way to choose", Operational Research, Tutorial papers, pages 53-101, Birmingham, 1990.

[BENOIT 98] M. BENOIT, G. CHICOISNE, J.-P. DEFFONTAINES, D. HERVE, S. LARDON, F. LE BER, C. MULLON, F. PAPY, V. SOUCHERE, P. THINON, M. TICHIT et J.-P. TREUIL: "Coordonner des choix de cultures sous contraintes environnementales : des jeux de rôle aux modèles multi-agents", Actes du Colloque SMAGET, pages 133-141, 1998.

[BERCHER 98] P. BERCHER : "Les catastrophes naturelles : inondations et coulées boueuses en Haute-Normandie, Rouen", Rapport de l'Agence Régionale de l'Environnement de Haute-Normandie (AREHN: http://www.arehn.asso.fr), 1998.

[BERTSIMAS 97] D. BERTSIMAS et J.N. TSITSIKLIS: "Introduction to linear optimization", Athena Scientific, Belmont, 1997.

[BESSIERE 94] C. BESSIERE: "Arc consistency and arc-consistency again", Artificial Intelligence, pages179-190, 1994.

[BESSIERE 95] C. BESSIERE, E. FREUDER et J.C. REGIN: "Using inference to reduce arc consistency computation", Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'95), pages 592-598, Canada, 1995.

[BESSIERE 99] C. BESSIERE: "Non-binary constraints", Proceedings of Principles and Practice of Constraint Programming, pages 24-27, USA, 1999.

[BESSIERE 01] C. BESSIERE et J.C. REGIN: "Refining the basic constraint propagation algorithm", Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'01), pages 309-315, USA, 2001.

[BISTARELLI 97] S. BISTARELLI, U. MONTANARI et F. ROSSI: "Semiring-based constraint solving and optimisation", Association for Computing Machinery (ACM), pages 201-236, 1997.

[BISTARELLI 99] S. BISTARELLI, H. FARGIER, U. MONTANARI, F. ROSSI, T. SCHIEX et G. VERFAILLIE : "Semiring-based CSPs and Valued CSPs: Frameworks, properties, and comparison", Constraints, pages 199-240, 1999.

[BITNER 75] J. BITNER et E. M. REINGOLD: "Backtrack programming techniques", Association for Computing Machinery (ACM), pages 651-656, 1975.

[BLANCHARD 99] E. BLANCHARD, C. KING, Y. LE BISSONNAIS, A. BOURGUIGNON, V. SOUCHERE, J-F. DESPRATS et P. MAURIZOT: "Paramétrisation du potentiel de ruissellement des bassins versants au moyen de la Télédétection et des Systèmes d'Informations Géographiques, Application à des bassins versants du Pays de Caux", Etude et Gestion des Sols, pages 181-199, 1999.

[BOIFFIN 88] J. BOIFFIN, F. PAPY et M. EIMBERCK: "Influence des systèmes de culture sur les risques d'érosion", Agronomie, pages 663-673, 1988.

[BOLI 98] Z. BOLI, E. ROOSE et P. ZAHONERO: "Influences du système de culture sur les risques érosifs et la production intensive de coton et maïs en savane humide du Nord-Cameroun", Orstom actualités, pages 27-28, 1998.

[BONOMI 84] E. BONOMI et J.L. LUTTON: "The N-city travelling salesman problem", Statistical Mechanics and the Metropolis Algorithm, SIAM Review, pages 551-568, 1984.

[BOOCH 99] G. BOOCH, J. RUMBAUGH et I. JACOBSON: "The unified modelling language reference manual", Addison-Wesley, 1999.

[BORNING 87] A. BORNING, R. DUISBERG, B. FREEMAN-BENSON, K. KRAMER et M. WOLF: "Constraint hierarchies", Proceedings of ACM Conference on Object Oriented Programming Systems, Languages and Applications, pages 48-60, USA, 1987.

[BOU KHEIR 01] R. BOU KHEIR, M-CI. GIRARD, M.KHAWLIE et C. ABADALLAH: "Erosion hydrique des sols dans les milieux méditerranéens : Une revue bibliographique", Etude et Gestion des Sols, pages 231-245, 2001.

[BOUSQUET 01] F. BOUSQUET: "Modélisation d'accompagnement : Systèmes multi-agents et gestion des ressources naturelles et renouvelables", Mémoire pour l'obtention de l'Habilitation à Diriger les Recherches, Université de Lyon 1, 2001.

[BOUYSSOU 84] D. BOUYSSOU: "Approches descriptives et constructives d'aide à la décision: Fondements et comparaison", Thèse de doctorat, Université Paris-Dauphine, 1984.

[BOUYSSOU 86] D. BOUYSSOU: "Some remarks on the notion of compensation in MCDM", Operational Research, pages 150-160, 1986.

[BOUYSSOU 93] D. BOUYSSOU: "Décision multicritère ou aide multicritère?", Newsletter of the European Working Group "Multicriteria Aid for Decisions", pages 1-2, Spring, 1993.

[BRUYNOOGHE 84] M. BRUYNOOGHE et L. M. PEREIRA: "Deduction revision by intelligent backtracking", Implementations of Prolog, pages 194-215, 1984.

[BURA 91] S. BURA, A. DROGOUL, J. FERBER et E. JACOPIN: "Eco-résolution : Un modèle de résolution de problèmes par interactions", Actes du 8ème congrès AFCET-RFIA, pages 1299-1308, 1991.

[BURROUGH 86] P.A. BURROUGH: "Principles of geographical information systems for land resources assessment", Oxford University Press, USA, 1986.

[CELIK 96] I. CELIK, M. AYDIN et U. YAZICI: "A review of the erosion control studies during the republic period in Turkey", Proceedings of 1st International Conference on Land Degradation, pages 175-180, Turquie, 1996.

[CERDAN 01] O. CERDAN, V. SOUCHÈRE, V. LECOMTE, A. COUTURIER et Y. LE BISSONNAIS: "Incorporating soil surface crusting processes in an expert-base runoff model: Sealing and transfer by runoff and erosion related to agricultural management", Catena, n° 46, pages 189-205, 2001.

[CHARON 95] I. CHARON et O. HUDRY: "Mixing different component of metaheuristics", Metaheuristics, Theory and Applications, Kluwers Academic Publishers, pages 589-604, 1995.

[CHEBBANI 99] R. CHEBBANI, K. DJILLI et E. ROOSE: "Etude des risques d'érosion dans le bassin versant de l'isser, Algérie", Bulletin Réseau Erosion, pages 85-95, 1999.

[CHEN 99] F. CHEN, E.T. KANEMASU, L.T. WEST et F. RACHIDI: "Analysis of land use and simulation of soil erosion with GIS for the semi-arid region of Morocco", Géo-observateur, pages 55-75, 1999.

[CODOGNET 95] P. CODOGNET: "Programmation logique avec contraintes : Une introduction", Technique et Science Informatique, pages 662-692, 1995.

[COLLINS 88] N.E. COLLINS, R.W. EGLESE et B.L. GOLDEN: "Simulated annealing: an annotated bibliography", Mathematical and Management Sciences, pages 209-307, 1988.

[COLMERAUER 90] A. COLMERAUER: "An introduction to Prolog III", Communications of the Association for Computing Machinery (ACM), pages 69-90, 1990.

[CONNOLLY 90] D. CONNOLLY: "An improved annealing scheme for the QAP", Operational Research, pages 93-100, 1990.

[COURBON 93] J.C., COURBON, D. DUBOIS et B. ROY: "Autour de l'aide à la décision et de l'intelligence artificielle", Interrogés par J.C. POMEROL, Conférence AFCET, France, 1993.

[DAVIS 91] L. DAVIS: "Handbook of genetic algorithms", Van Nostrand Reinhold, USA, 1991.

[DE KLEER 86a] J. DE KLEER: "An assumption-based TMS", Artificial Intelligence, pages 127-162, 1986.

[DE KLEER 86b] J. DE KLEER: "Problems with ATMS", Artificial Intelligence, pages 197-224, 1986.

[DE ROO 93] A.P.J. DE ROO: "Modelling surface runoff and soil erosion in catchments using geographical information systems, validity and applicability of the "ANSWERS" model in two catchments in the loess area of South Limbourg and one in Devon", Thèse de doctorat, Université d'Utrecht, 1993.

[DECHTER 86] R. DECHTER: "Learning while searching in constraint satisfaction problems", Proceedings of National Conference on Artificial Intelligence, American Association for Artificial Intelligence (AAAI'86), pages 178-185, USA, 1986.

[DECHTER 89] R. DECHTER et I. MEIRI: "Experimental evaluation of pre-processing techniques in constraint-satisfaction problems", Proceedings of the 11th International Joint Conference on Artificial Intelligence, pages 290-296, USA, 1989.

[DECHTER 90] R. DECHTER: "Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition", Artificial Intelligence, pages 273-312, 1990.

[DEGOULET 91] P. DEGOULET et M. FIESCHIE: "Traitement de l'information médicale : méthodes et applications hospitalières", Editions Masson, paris, 1991.

[DELAHAYE 92] D. DELAHAYE: "Approches spatialisées et analyses expérimentales des phénomènes de ruissellement et d'érosion des sols : Application aux systèmes de production agricole du Calvados", Thèse de doctorat, Université de Caen, 1992.

[DELAHAYE 95] D. DELAHAYE et A.E. HAUCHARD: "L'inondation de Grainville-la-Teinturière, analyse des processus de ruissellement en Pays de Caux au travers d'un épisode critique", La Revue d'Ici, pages 12-17, 1995.

[DELAHAYE 98] D. DELAHAYE et E. HAUCHARD: "Analyse spatiale des processus de ruissellement en Pays de Caux au travers de quelques épisodes critiques", Bulletin de l'Association des Géographes Français (AGF), pages 306-316, 1998.

[DELAHAYE 99a] D. DELAHAYE: "Originalité des risques hydrologiques en Seine-Maritime : La catastrophe de Saint-Martin-de-Boscherville en juin 1997", Etudes Normandes, pages 157-170, 1999.

[DELAHAYE 99b] D. DELAHAYE, E. HAUCHARD et D GAILLARD: "Analyse des processus de ruissellement et d'inondation dans le pays de Caux (France), intérêt d'une approche géomorphologique", Paysages Agraires et Environnement, pages 209-219, CNRS Editions, 1999.

[DELAHAYE 99c] D. DELAHAYE, Y. GUERMOND, P. FOLLIGNE, J-P. VAPAILLE et R. ZAMBEAUX: "Une recherche sur la pollution des sols de la rive gauche à Rouen", Etudes Normandes, pages179-186, 1999.

[DENEGRE 96] J. DENEGRE et F. SALGE: "Les systèmes d'information géographique", Presses Universitaires de France, Paris, 1996.

[D'HAESELEER 94] P. D'HAESELEER: "Context preserving crossover in genetic programming", Proceedings of the IEEE World Congress on Computational Intelligence, pages 256-261, USA, 1994.

[DEVILLE 91] Y. DEVILLE et P.V. HENTENRYCK: "An efficient arc consistency algorithm for a class of CSP problems", Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91), pages 325-330, Australie, 1991.

[DINCBAS 88] M. DINCBAS, P. VAN HENTENRYCK, H. SIMONIS, A. AGGOUN, T. GRAF et F. BERTHIER: "The constraint logic programming language CHIP", Proceedings of the International Conference on Fifth Generation Computer Systems, pages 693-702, Japan, 1988.

[DORIGO 99] M. DORIGO et G. DI CARO: "The Ant colony optimization meta-heuristic", New Ideas in Optimization, pages 11-32, 1999.

[DUBOIS 95] D. DUBOIS, H. FARGIER et H. PRADE: "Fuzzy constraints in job-shop scheduling", Intelligent Manufacturing, pages 215-234, 1995.

[DUCEK 90] G. DUCEK et T. SCHEUER: "Threshold accepting: A general purpose optimization algorithm", Computational Physics, pages 161-175, 1990.

[DUCEK 93] G. DUCEK: "New optimization heuristics: The great deluge algorithm and the record-to-record travel", Computational Physics, pages 86-92, 1993.

[EASTMAN 72] C.M. EASTMAN: "Preliminary report on a system for general space planning", Communications of the Association for Computing Machinery (ACM), pages 76-87, 1972.

[EASTMAN 99] J.R. EASTMAN: "Multi-criteria evaluation and GIS", Geographical Information Systems, pages 493-502, 1999.

[EL MAGNOUNI 93] S. EL MAGNOUNI: "Méthodologie d'aide à la décision pour l'évaluation et la gestion multicritère des ressources en eau souterraine", Thèse de Doctorat en Génie Civil et Minier, Institut National Polytechnique de Lorraine, Nancy, 1993.

[FARGIER 93a] H. FARGIER, J. LANG et T. SCHIEX: "Selecting preferred solutions in fuzzy constraint satisfaction problems", Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies (EUFIT), pages 128-134, Allemagne, 1993.

[FARGIER 93b] H. FARGIER et J. LANG: "Uncertainty in constraint satisfaction problems: A probabilistic approach", Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning about Uncertainty (ECSQARU'93), pages 97-104, Espagne, 1993.

[FARGIER 96] H. FARGIER, J. LANG et T. SCHIEX: "Mixed constraint satisfaction: a framework for decision problems under incomplete knowledge", Proceedings of National Conference on Artificial Intelligence, American Association for Artificial Intelligence (AAAI'96), pages 175-180, USA, 1996.

[FEO 89] T.A. FEO et M.G.C. RESENDE: "A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters, pages 67-71, 1989.

[FEO 94] T.A. FEO, M.G.C. RESENDE et S.H. SMITH: "A greedy randomized adaptive search procedure for maximum independent set", Operations Research, pages 860-878, 1994.

[FEO 95] T.A. FEO et M.G.C. RESENDE: "Greedy randomized adaptive search procedures", Global Optimization, pages 109-133, 1995.

[FOX 87] M.S. FOX: "Constraint-directed search: A case study of job shop scheduling", Pitman Publishers, London, 1987.

[FOX 89] M.S. FOX, N. SADEH et C. BAYKAN: "Constrained heuristic search", Proceedings of the 11th International Joint Conference on Artificial Intelligence, pages 309-315, USA, 1989.

[FOGEL 92] D.B. FOGEL, L.J. FOGEL, J.W. ATMAR et G.B. FOGEL: "Hierarchic methods of evolutionary programming", Proceedings of the 1st Annual Conference on Evolutionary Programming, pages 175-182, USA, 1992.

[FREEMAN 90] B.N. FREEMAN-BENSON, J. MALONEY et A. BORNING: "An incremental constraint solver", Communications of the Association for Computing Machinery (ACM), pages 54-63, 1990.

[FREUDER 85] E. FREUDER et M. QUINN: "Taking advantage of stable sets of variables in constraint-satisfaction problems", Proceedings of the 9th International Joint Conference on Artificial Intelligence, pages 1076-1078, USA, 1985.

[FREUDER 89] E. C. FREUDER: "Partial constraint satisfaction", Proceedings of 11th International Joint Conference on Artificial Intelligence, pages 278-283, USA, 1989.

[FREUDER 92] E.C. FREUDER et R.J. WALLACE: "Partial constraint satisfaction", Artificial Intelligence, pages 21-70, 1992.

[GAILLARD 02] D. GAILLARD, D. DELAHAYE, P. LANGLOIS et W. JAZIRI: "L'automate cellulaire comme outil de cartographie de la dynamique du ruissellement en territoire agricole : Cas du Pays de Caux, Seine-Maritime, France", Proceedings of the International Symposium of Geomorphology: From Expert Opinion to Modelling, pages 321-324, Strasbourg, France, 2002.

[GALLIEN 95] E. GALLIEN, Y. LE BISSONNAIS, M. EIMBERCK, H. BENKHADRA, L. LIGNEAU, J.F. OUVRY et P. MARTIN: "Influence des couverts végétaux de jachère sur le ruissellement et l'érosion diffuse en sol limoneux cultivé", Cahiers Agricultures, pages 171-183, 1995.

[GAREY 79] M.R. GAREY et D.S. JOHNSON: "Computers and intractability: A guide to the theory of NP-completeness", W.H. Freeman and Company, USA, 1979.

[GARY 93] G. GARY: "Le risque d'inondation en France", Thèse de doctorat, Université de Paris I, 1993.

[GASCHNIG 78] J. GASCHNIG: "Experimental case studies of backtrack versus waltz-type versus new algorithms for satisfying assignment problems", Proceedings of the 2nd Biennial Conference of the Canadian Society for Computational Studies of Intelligence, pages 268-277, Canada, 1978.

[GASCHNIG 79] J. GASCHNIG: "Performance measurement and analysis of certain search algorithms", Thèse de doctorat, Department of Computer Science, Carnegie Mellon University, CMU-CS-79-124, USA, 1979.

[GERSHON 84] M. GERSHON et L. DUCKSTIEN: "A procedure for selection of a multi-objective technique with application to water and mineral resources", Applied Mathematics and Computation, pages 245-271, 1984.

[GLOVER 86] F. GLOVER: "Future paths for integer programming and links to artificial intelligence", Computers and Operations Research, pages 533-549, 1986.

[GLOVER 95] F. GLOVER, J.P. KELLY et M. LAGUNA: "Genetic algorithms and tabu search: Hybrids for optimization", Computers and Operations Research, pages 111-134, 1995.

[GLOVER 97] F. GLOVER et M. LAGUNA: "Tabu Search", Kluwer Academic Publishers, 1997.

[GLOVER 00] F. GLOVER, M. LAGUNA et R. MARTI: "Fundamentals of scatter search and path relinking", Control and Cybernetics, pages 653-684, 2000.

[GOLDBERG 89] D.E. GOLDBERG: "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley, USA, 1989.

[GRANGER 02] G. GRANGER: "Détection et résolution de conflits aériens : Modélisations et analyse", Thèse de doctorat, Ecole Polytechnique, France, 2002.

[GREFENSTETTE 87] J.J. GREFENSTETTE: "Incorporating problem specific knowledge into genetic algorithms", Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, pages 42-60, 1987.

[GU 89] J. GU: "Parallel algorithms and architectures for very fast AI search", Thèse de doctorat, Department of Computer Science, Université de Utah, USA, 1989.

[HAJEK 88] B. HAJEK: "Cooling schedules for optimal annealing", Mathematics of Operations Research, pages 311-329, 1988.

[HAO 99] J-K. HAO, P. GALINIER et M. HABIB: "Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes", Intelligence Artificielle, pages 283-324, 1999.

[HARALICK 80] R. HARALICK et G. ELLIOT: "Increasing tree search efficiency for constraint satisfaction problem", Artificial Intelligence, pages 263-313, 1980.

[HOLLAND 75] J.H. HOLLAND: "Adaptation in natural and artificial systems", The University of Michigan Press, Ann Arbor, 1975.

[HOLLAND 92] J.H. HOLLAND: "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence", MIT Press/Bradford Books, 1992.

[JACQUET 82] E. JACQUET-LAGRÈZE et J. SISKOS: "Assessing a set of additive utility functions for multicriteria decision-making, the UTA method", Operational Research, pages 151-164, 1982.

[JAZIRI 02a] W. JAZIRI, T. PAQUET, A. ALIMI et D. DELAHAYE: "Multi-level optimization strategy for the prevention off run-off risk", Proceedings of IEEE-SMC'02, volume 1, pages 466-471, Tunisie, 2002.

[JAZIRI 02b] W. JAZIRI, T. PAQUET, D. GAILARD et A. ALIMI: "Knowledge modelling and multi-agent simulation: Application to flood risks", Proceedings of IEEE-SMC'02, 6 pages sur CDRom, code papier : TA1O3, Tunisie, 2002.

[JANSSEN 90] R. JANSSEN et P. RIETVELD: "Multicriteria analysis and Geographic Information Systems: An application to agricultural land use in the Netherlands", Geographical Information Systems for Urban and Regional Planning, pages 129-139, Dordecht, Kluwer, 1990.

[JETTEN 96] V. JETTEN, J. BOIFFIN et A. DE ROO: "Defining monitoring strategies for runoff and erosion studies in agricultural catchments: A simulation approach", Soil Science, volume 47, pages 579-592, 1996.

[JOERIN 97] F. JOERIN: "Décider sur le territoire, proposition d'une approche par utilisation de SIG et de méthodes d'analyse multicritère", Thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, 1997.

[JUSSIEN 97] M.N. JUSSIEN: "Relaxation de contraintes pour les problèmes dynamiques", Thèse de doctorat, Université de Rennes 1, France, 1997.

[KEENEY 76] R.L. KEENEY et H. RAIFFA: "Decisions with multiple objectives: Preferences and value tradeoffs", Wiley and Sons, USA, 1976.

[KERNIGHAN 70] B.W. KERNIGHAN et S. LIN: "An efficient heuristic for partitioning graphs", Bell System Technology Journal, pages 291-307, 1970.

[KING 92] D. KING, Y. LE BISSONNAIS, R. HARDY et M. EIMBERCK: "Combinaison spatiale d'informations pour l'évaluation des risques de ruissellement à l'échelle régionale", Gestion de l'Espace Rural et Système d'Information Géographique, pages 149-166, 1992.

[KIRKPATRICK 83] S. KIRKPATRICK, C.D. GELATT et P.M. VECCHI: "Optimization by simulated annealing", Science, pages 671-680, 1983.

[KOULAMAS 94] C. KOULAMAS, S.R. ANTHONY et R. JEAN: "A survey of simulated annealing: application to operations research problems", Omega, pages 41-56, 1994.

[KUMAR 87] V. KUMAR: "Depth-first search", Artificial Intelligence, pages 1004-1005, USA, 1987.

[KUMAR 88] V. KUMAR et Y.J. LIN: "A data-dependency-based intelligent backtracking scheme for prolog", Logic Programming, pages 165-181, 1988.

[KUMAR 92] V. KUMAR: "Algorithms for constraint satisfaction problems: A survey", Artificial Intelligence, pages 32-44, 1992.

[LAARIBI 95] A. LAARIBI: "Systèmes d'information géographique et analyse multicritère : Intégration pour l'aide à la décision à référence spatiale", Thèse de doctorat, Faculté de Foresterie et de Géomatique, Université Laval, Québec, 1995.

[LAGUNA 02] M. LAGUNA: "Scatter search", In Handbook of Applied Optimization, Oxford University Press, pages 183-193, 2002.

[LANGLOIS 94] P. LANGLOIS: "Formalisation des concepts topologiques en géomatique", Géomatique, pages 181-205, Hermès, 1994.

[LANGLOIS 97] A. LANGLOIS et M. PHIPPS: "Automates cellulaires: Application à la simulation urbaine", Hermès, 1997.

[LANGLOIS 02] P. LANGLOIS et D. DELAHAYE: "RuiCells, Automate Cellulaire pour la Simulation du Ruissellement de Surface", Géomatique, Hermès, pages 461-487, 2002.

[LARDON 92] S. LARDON: "SIG: Nouveaux concepts pour des démarches nouvelles", Gestion de l'Espace Rural et Système d'Information Géographique, pages 67-77, 1992.

[LAURIERE 78] J.L. LAURIERE: "A language and a program for stating and solving combinatorial problems", Artificial Intelligence, pages 29-127, 1978.

[LAWLER 96] E.L. LAWLER et D.E. WOOD: "Branch and bound methods: A survey", Operations Research, pages 699-719, 1996.

[LEBER 98a] F. LE BER, A. DURY, V. CHEVRIER et M. BENOIT: "Simuler l'organisation spatiale d'un territoire agricole : Différentes approches", Actes du Colloque SMAGET, Cemagref, Clermont Ferrand, pages 249-259, 1998.

[LEBER 98b] F. LE BER, V. CHEVRIER et A. DURY: "A multi-agent system for the simulation of land use organisation", Proceedings of 3rd IFAC/CIGR Workshop on Artificial Intelligence in Agriculture, pages 182-187, Japon, 1998.

[LEBER 98c] F. LE BER et M. BENOÎT: "Modelling the spatial organisation of land use in a farming territory, example of a village in the Plateau Lorrain", Agronomie: Agriculture and Environment, pages 101-113, 1998.

[LE BISSONNAIS 96] Y. LE BISSONNAIS, H. BENKHADRA, E. GALLIEN, M. EIMBERCK, D. FOX, P. MARTIN, C. DOUYER, L. LIGNEAU et J.F. OUVRY: "Genèse du ruissellement et de l'érosion diffuse sur sols limoneux : Analyse du transfert d'échelle du m² au bassin versant élémentaire agricole", Géomorphologie : Relief, Processus, Environnement, pages 51-64, 1996.

[LE BISSONNAIS 97] Y. LE BISSONNAIS et F. PAPY: "Les effets du ruissellement et de l'érosion sur la turbidité de l'eau", L'eau dans l'agro-écosystème, pages 323-338, INRA Editions, France, 1997.

[LECHEVALIER 91] C. LECHEVALIER: "L'érosion des terres agricoles en Pays de Caux", Etudes normandes, pages 97-116, 1991.

[LEGEARD 00] B. LEGEARD: "Vers un outil d'aide à la décision spatialisée pour la gestion des inondations torrentielles : Le système MAGIC", Cemagref Editions, pages 379-391, Antony, France, 2000.

[LIN 73] S. LIN et B.W. KERNIGHAN: "An efficient heuristic for the travelling-salesman problem", Operations Research, pages 498-516, 1973.

[LIVADAS 00] C. LIVADAS, J. LYGEROS et N.A. LYNCH: "High-Level modeling and analysis of the Traffic alert and Collision Avoidance System (TCAS)", Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications, pages 926-948, 2000.

[LUNDY 86] S. LUNDY et A. MEES: "Convergence of an annealing algorithm", Mathematical Programming, pages 111-124, 1986.

[MACKWORTH 77] A.K. MACKWORTH: "Consistency in networks of relations", Artificial Intelligence, pages 99-118, 1977.

[MACKWORTH 85] A.K. MACKWORTH, J.A. MULDER et W.S. HAVENS: "Hierarchical arc consistency: Exploiting structured domains in constraint satisfaction", Computational Intelligence, pages 118-126, 1985.

[MACKWORTH 92] A.K.MACKWORTH: "Constraint satisfaction", Artificial Intelligence, pages 285-293, Wiley-Interscience, USA, 1992.

[MARTIN 97] P. MARTIN, Y. LE BISSONNAIS, H. BENKHADRA, L. LIGNEAU et J.F. OUVRY: "Mesures du ruissellement et de l'érosion diffuse générés par les pratiques culturales en Pays de Caux (Normandie)", Géomorphologie: Relief, Processus, Environnement, pages 143-154, 1997.

[MARTIN 98] P. MARTIN, F. PAPY, V. SOUCHERE et A. CAPILLON: "Maîtrise du ruissellement et modélisation des pratiques de production", Cahiers Agricultures, pages 111-119, 1998.

[MARTIN 99] P. MARTIN: "Reducing flood risk from sediment-laden agricultural runoff using intercrop management techniques in Northern France", Soil and Tillage Research, pages 233-245, 1999.

[MARTIN 01] P. MARTIN, F. PAPY et A. CAPILLON: "Agricultural field state and runoff Risk: Proposal of a simple relation for the silty-loam-soil context of the Pays de Caux (France)", Proceedings of the 10th International Soil Conservation Organization Meeting, In Sustaining the Global Farm, pages 293-299, 2001.

[MARTEL 93] J.M. MARTEL et A. ROUSSEAU: "Cadre de référence d'une démarche multicritère de gestion intégrée des ressources en milieu forestier", Projet de développement de la gestion intégrée des ressources, document technique 93/11, Université Laval, Québec, 1993.

[MAYSTRE 94] L. MAYSTRE, J. PICTET et J. SIMOS: "Méthodes multicritère ELECTRE : Description, conseils pratiques et cas d'application à la gestion environnementale", Presses Polytechniques et Universitaires Romandes, CH-1015 Lausanne, Suisse, 1994.

[McGREGOR 79] J. McGREGOR: "Relational consistency algorithms and their applications in finding subgraph and graph isomorphism", Information Science, pages 229-250, 1979.

[MILLER 86] P. MILLER: "Expert critiquing systems: Practice-based medical consultation by computer", Springer-Verlag, USA, 1986.

[MINTON 92] S. MINTON, M.D. JOHNSTON, A.B. PHILIPS et P. LAIRD: "Minimizing conflicts: A heuristics repair method for constraint satisfaction and scheduling problems", Artificial Intelligence, pages 161-205, 1992.

[MITTAL 87] S. MITTAL et F. FRAYMAN: "Making partial choices in constraint-reasoning problems", Proceedings of the 6th National Conference on Artificial Intelligence, pages 631-636, USA, 1987.

[MOHR 86] R. MOHR et T. HENDERSON: "Arc and path consistency revisited", Artificial Intelligence, pages 225-233, 1986.

[MUSY 03] A. MUSY: "Cours hydrologie générale 2003", Laboratoire d'Hydrologie et Aménagements (HYDRAM), Institut des Sciences et Technologies de l'Environnement (ISTE), Ecole Polytechnique Fédérale de Lausanne, Site web : http://hydram.epfl.ch/e-drologie.

[NADEL 88] B. NADEL: "Tree search and arc consistency in constraint-satisfaction algorithms", Search in Artificial Intelligence, pages 287-342, Springer-Verlag, USA, 1988.

[OTHMANI 98] I. OTHMANI: "Optimisation multicritère : Fondements et concepts", Thèse de doctorat de l'Université Joseph Fourier-Grenoble 1, France, 1998.

[PAPADIMITRIOU 82] C.H. PAPADIMITRIOU et K. STEIGLITZ: "Combinatorial optimization: Algorithms and complexity", Prentice Hall, 1982.

[PAPARRIZOS 03] K. PAPARRIZOS, N. SAMARAS et G. STEPHANIDES: "A new efficient primal dual simplex algorithm", Computers and Operations Research, pages 1383-1399, 2003.

[PAPY 91] F. PAPY et C. DOUYER: "Influence des états de surface du territoire agricole sur le déclenchement des inondations catastrophiques", Agronomie, pages 201-215, 1991.

[PAQUET 01] T. PAQUET, W. JAZIRI et P. LANGLOIS: "Simulation multi-échelles pour l'analyse des risques de ruissellement", Actes des journées Cassini, pages 295-296, France, 2001.

[PEARL 84] J. PEARL: "Heuristics", Addison-Wesley, 1984.

[PEREIRA 93] J. M.C. PEREIRA et L. DÜCKSTEIN: "A multiple criteria decision-making approach to GIS-based land suitability evaluation", Geographical Information Systems, pages 407-424, 1993.

[PRANDINI 99] M. PRANDINI, J. LYGEROS, A. NILIM et S. SASTRY: "A probabilistic framework for aircraft conflict detection", Proceedings of the Guidance, Navigation and Control Conference, American Institute of Aeronautics and Astronautics (AIAA), pages 1047-1057, USA, 1999.

[PRINCE 96] V. PRINCE: "Vers une informatique cognitive dans les organisations : Le rôle central du langage", Masson, France, 1996.

[PROSSER 91] P. PROSSER: "Hybrid algorithms for the constraint-satisfaction problem", Technical Report, AISL-46-91, Department of Computer Science, Université de Strathclyde, Scotland, 1991.

[PROSSER 95] P. PROSSER: "MAC- CBJ: Maintaining arc-consistency with conflict-directed backjumping", Technical Report 95/177, Department of Computer Science, Université de Strathclyde, Scotland, 1995.

[PUGET 94] J.P. PUGET: "A C++ implementation of CLP", Proceedings of 2nd International Conference on Intelligent Systems, pages 256-261, Singapore, 1994.

[ROOSE 92] E. ROOSE E., P. DUGUE et L. RODRIGUEZ: "La GCES : une nouvelle stratégie de lutte antiérosive appliquée à l'aménagement de terrasses en zone soudano-sahélienne du Burkina Faso", Bois et Forêts des Tropiques, pages 49-63, 1992.

[ROSENFELD 76] A. ROSENFELD, R. HUMMEL et S. ZUCKER: "Scene labeling by relaxation operations", IEEE Transactions on Systems, Man, and Cybernetics, pages 173-184, 1976.

[ROY 68] B. ROY: "Classement et choix en présence de points de vue multiples (la méthode Electre)", Recherche Opérationnelle, pages 57-75, 1968.

[ROY 75] B. ROY: "Vers une méthodologie générale d'aide à la décision", Metra, pages 459-497, 1975.

[ROY 81] B. ROY: "The optimisation problem formulation: Criticism and overstepping", The Operational Research Society, pages 427-436, 1981.

[ROY 85] B. ROY: "Méthodologie multicritère d'aide à la décision", Economica, 1985.

[ROY 90] B. ROY: "Science de la décision ou science de l'aide à la décision?", Cahier du Lamsade 97, Université Paris-Dauphine, France, 1990.

[ROY 92] B. ROY, R. SLOWINSKI et W. TREICHEL: "Multicriteria programming of water supply systems for rural areas", Water Resources Bulletin, pages 13- 31, 1992.

[ROY 93] B. ROY et D. BOUYSSOU: "Aide multicritère à la décision: Méthodes et cas", Economica, Paris, 1993.

[ROY 96] B. ROY: "Multicriteria methodology for decision aiding", Kluwer, 1996.

[RUSTICHINI 98] A. RUSTICHINI: "Dynamic programming solution of incentive constrained problems", Economic Theory, pages 329-354, 1998.

[SHAPIRO 81] L. SHAPIRO et R. HARALICK: "Structural descriptions and inexact matching", IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 504-518, Israel, 1981.

[SCHÄRLIG 85] A. SCHÄRLIG: "Décider sur plusieurs critères, panorama de l'aide à la décision multicritère", Presses Polytechniques et Universitaires Romandes, Suisse, 1985.

[SCHIEX 92] T. SCHIEX: "Possibilistic constraint satisfaction problems or "How to handle soft constraints?"", Proceedings of the 8th International Conference on Uncertainty in Artificial Intelligence (UAI-92), pages 268-275, USA, 1992.

[SCHIEX 95] T. SCHIEX, H. FARGIER et G. VERFAILLIE: "Valued constraint satisfaction problems: Hard and easy problems", Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), pages 631-637, 1995.

[SCHIEX 96] T. SCHIEX., J.-C. REGIN, C. GASPIN et G. VERFAILLIE: "Lazy arc consistency", Proceedings of the 13th National Conference on Artificial Intelligence, pages 216-221, USA, 1996.

[SCHIEX 97] T. SCHIEX, H. FARGIER et G. VERFAILLIE: "Problèmes de satisfaction de contraintes valués", Intelligence Artificielle, pages 339-373, 1997.

[SEFION 02] H. SEFION, M. GAILHARDOU et A. ENNAJI: "A medical decision support system for asthmatic patient health care", International Conference on Artificial Intelligence and Application (AIA'02), pages 83-86, Espagne, 2002.

[SMITH 00] B. SMITH, K. STERGIOU et T. WALSH: "Using auxiliary variables and implied constraints to model non-binary problems", Proceedings of National Conference on Artificial Intelligence, American Association for Artificial Intelligence (AAAI'00), USA, 2000.

[SOLAND 79] R.M. SOLAND: "Multicriteria optimization: A general characterization of efficient solutions", Decision sciences, pages 26-38, 1979.

[SOUCHERE 95] V. SOUCHERE: "Modélisation spatiale du ruissellement à des fins d'aménagement contre l'érosion de talweg, application à des petits bassins versants en Pays de Caux (Haute Normandie)", Thèse de doctorat, Institut National Agronomique Paris-Grignon, France, 1995.

[STALLMAN 77] R. STALLMAN et G.J. SUSSMAN: "Forward reasoning and dependency-directed backtracking", Artificial Intelligence, pages 135-196, 1977.

[STERGIOU 99] K. STERGIOU et T. WALSH: "Encodings of non-binary constraint satisfaction Problems", Proceedings of National Conference on Artificial Intelligence, American Association for Artificial Intelligence (AAAI'99), pages 163-168, USA, 1999.

[STEUER 86] R.E. STEUER: "Multiple criteria optimization: Theory, computation, and application", Wiley, USA, 1986.

[STROMBONI 85] J.P. STROMBONI: "On man-machine interface adaptation", Proceedings of Applied Ergonomics, pages 33-44, Danemark, 1985.

[TANGUIANE 91] A.S. TANGUIANE: "Aggregation and representation of preferences", Springer-Verlag, Berlin, 1991.

[TECLE 91] A. TECLE et L. DUCKSTEIN: "Concepts on multicriterion decision making", Decision Support Systems in Water Resources Management, pages 33-62, UNESCO Press, Paris, 1994.

[TIM 94] U.-S. TIM et R. TOLLY: "Evaluating agricultural non-point source pollution using integrated geographic information systems and hydrologic/water quality model", Environment Quality, pages 25-35, 1994.

[T'KINDT 02] V. T'KINDT et J.-C. BILLAUT: "Multicriteria scheduling: Theory, models and algorithms", Springer, Allemagne, 2002.

[TSANG 93] E.P.K. TSANG: "Foundations of constraint satisfaction", Academic press, London, 1993.

[TVERSKY 69] A. TVERSKY: "Intransitivity of preferences", Psychological Review, pages 31-48, 1969.

[VANSNICK 86] J.C. VANSNICK: "On the problem of weights in multiple criteria decision making: The non compensatory approach", Operational Research, pages 288-294, 1986.

[VIJAYALAKSHMI 87] B. VIJAYALAKSHMI: "An application of multi-objective modeling: The case of the Indian sugar industry", Operational Research, pages 146-153, 1987.

[VINCE 02] A. VINCE: "A framework for the greedy algorithm", Discrete Applied Mathematics, pages 247-260, 2002.

[VINCKE 82] P. VINCKE: "Aggregation of preferences: A review", Operational Research, pages 17-22, 1982.

[VINCKE 89] P. VINCKE: "L'aide multicritère à la décision", Editions de l'Université de Bruxelles, 1989.

[WALTZ 75] D. WALTZ: "Understanding line drawings of scenes with shadows", The Psychology of Computer Vision, pages 19-91, Cambridge, 1975.

[WHITE 90] D.J. WHITE: "A Bibliography on the applications of mathematical programming multiple-objective methods", Operational Research Society, pages 669-691, 1990.

[WIRTH 86] N. WIRTH: "Algorithms and data structures", Prentice-Hall, USA, 1986.

ANNEXE

Sommaire

Annexe 1 :

Algorithmes de résolution

Algorithme du Backtrack :

Soit V_I l'ensemble des variables instanciées et V_N l'ensemble des variables non instanciées ;

Procédure Backtrack (V_I, variableCourante, V_N, uneValeur)

Début

- variableCourante = uneValeur
- *Si* l'instanciation courante est consistante *alors*
  - *Si* $V_N = \phi$ *alors* /* toutes les variables sont instanciées avec succès */
    - L'instanciation courante est une solution
  - *Sinon* // il reste des variables non instanciées
    - Soit variableSuivante : la prochaine variable à instancier
    - *Tant que* Dom (variableSuivante) non vide *faire*
      - Prendre une valeur v ∈ Dom (variableSuivante)
      - Backtrack ($V_I \cup$ {variableCourante},variableSuivante, V_N -{variableSuivante}, v)
    - *FinTant que*
  - *Finsi*
- *Finsi*

Fin

Algorithme de Recuit Simulé :

Début

- Soit une configuration initiale $\mathbf{x_i}$, une fonction de coût **f**, une température initiale élevée **T** et une fonction de réduction de température α.
- *Répéter*
  - Générer un voisin **x** par opération aléatoire sur $\mathbf{x_i}$.
  - $\mathbf{\Delta E} = \mathbf{f(x)} - \mathbf{f(x_i)}$: différence de coût // e.g., **f** : la valeur du risque de ruissellement.
  - *Si* $\mathbf{\Delta E} < 0$ *alors* $\mathbf{x_i} = \mathbf{x}$.
  - *Sinon*
    - Choisir r aléatoirement entre 0 et 1
    - *Si* $\mathbf{r} < \exp(\mathbf{-\Delta E / T})$ *alors* $\mathbf{x_i} = \mathbf{x}$.
  - Réduire la température $\mathbf{T = \alpha(T)}$ // Exemple $\mathbf{\alpha(T) = T \times r}$

- *Jusqu'à* condition d'arrêt (Si on n'a pas changé x durant un certain temps (pendant k itérations) ou si la simulation dure depuis assez longtemps ou si **T**=0).

Fin

Algorithmes Génétiques :

Début

- Générer aléatoirement une population initiale de **n** chromosomes (individus)
- Affecter une valeur de "fitness" à chaque individu de la population //**Evaluer l'adaptation**
- *Tant que* (il n'y a pas de solution satisfaisante) et (le temps est inférieur au temps limite) et (le nombre **c** de cycles est inférieur au nombre maximum fixé) *faire*
    - Incrémenter le temps; incrémenter **c** ;
    - *Répéter* // Dans chaque cycle, **n'** chromosomes sont choisis pour reproduction et croisement 2 à 2
        - Choisir 2 parents $\mathbf{p_1}$ et $\mathbf{p_2}$ en fonction de leur valeur de "fitness" ou aléatoirement **//Sélection**

        /* Si la sélection est aléatoire, une étape doit précéder la sélection pour reproduire chaque chromosome **i** de façon à augmenter la chance des mieux adaptés pour être sélectionnés */
        - Appliquer les opérateurs génétiques sur les copies des parents
            - Croiser $\mathbf{p_1}$ et $\mathbf{p_2}$ →2 enfants $\mathbf{e_1}$ et $\mathbf{e_2}$ **// Croisement**
            - Muter $\mathbf{e_1}$ et $\mathbf{e_2}$ avec une probabilité faible **Pm // Mutation**
        - Ajouter $\mathbf{e_1}$ et/ou $\mathbf{e_2}$ à la population précédente **// Gestion de la population**
    - *Jusqu'à* ce que **n** descendants aient été produits
    - Affecter une valeur de "fitness" à chaque individu de la population **// Evaluer l'adaptation**
    - Choisir les **n** meilleurs chromosomes
- *Fin Tant que*

Fin

Annexe 2 :

Grammaire des besoins

Nous présentons dans cette annexe la composition de la grammaire définie dans notre approche pour la saisie et la manipulation de la liste des contraintes (suivant la notation BNF) :

X={‘cut’, ‘select’, ‘evaluate’, ‘update’, ‘order’, ‘nom’, ‘niveau_préférence’, ‘complexité’, ‘poids’, ‘min’, ‘max’, ‘minimize’, ‘maximize’, ‘maintenir’ ‘∑’, ‘∏’, ‘moyenne’, ‘différence’, ‘<’, ‘>’, ‘<>’,’<=’, ‘>=’, ‘=’ , ‘ε’, ‘(‘, ‘)’, ‘,’, ’.’, ‘:’, ‘;’, '?', ‘$’, ‘+’, ‘-’, ‘_’, ‘!’, ‘\’, ‘/’, ‘~’, ‘\’, ‘*’, ‘%’, ‘#’, ‘"’, '0'..'9', 'a'..'z' , 'A'..'Z', ‘[’, ’]’, ‘{’, ‘}’, ‘Scénario’, ‘Besoin’}.

S={Intervention_acteur }.

N={Intervention_acteur, Expression_besoin, Suite_intervention, Scénario_liste, Suite_scénario, Opération_liste, Label_besoin, Label_scénario, Deux_points, Point_virgule, Action_liste, Action_simple, Action_composée, Evaluation, Classement, Modification, Mise_à_jour, Sélection, Tailler, Liste_contraintes, Contrainte, Champ_contrainte, Type_champ_contrainte, Valeur, Condition, Comparateur, Expression, Expression_simple, Suite_contrainte, Attribut, Suite_attribut, Comparant, Agrégat, Fonction_math, Borne, Sens_évolution, Paranthèse_ouvrante, Paranthèse_fermante, Point, Somme, Produit, Différence, Moyenne, Minimum, Maximum, Minimisation, Maximisation, Maintient, Entier, Réel, Chiffre, Virgule, Chaîne, Suite_Chaîne, Lettre, Guillemet}.

Règles de dérivation générant des variables :

Intervention_acteur :

<Intervention_acteur> :: = <Expression_besoin> <Suite_intervention>

Suite_intervention :

<Suite_intervention> :: = [<Scénario_liste>]

Expression_besoin :

<Expression_besoin> :: = [<Label_besoin> <Deux_points> <Liste_contraintes>]

Scénario_liste :

<Scénario_liste> :: = <Label_scénario> <Deux_points> <Opération _liste> <Suite_scénario>

Suite_scénario :

<Suite_scénario> :: = { <Point_virgule> <Opération _liste> <Suite_scénario> }

Opération_liste :

<Opération_liste> :: = <Action_liste> <Liste_contraintes>

Action_liste :

<Action_liste> :: = <Action_simple> | <Action_composée>

Action_simple :

<Action_simple> :: = <Evaluer> | <Classer>

Action_composée :

<Action_composée> :: = <Suppression> | <Modification> | <Sélection>

Suppression :

<Suppression> :: = <Tailler> <Liste_contraintes>

Modification :

<Modification> :: = <Mise_à_jour> <Liste_contraintes> <Champ_contrainte> <Valeur>

Sélection :

<Sélection> :: = <Sélectionner> <Condition>

Condition :

<Condition> :: = <Champ_contrainte> <Comparateur> <Type_champ_contrainte>

Type_champ_contrainte :

<Type_champ_contrainte> :: = <Valeur> | <Chaîne>

Liste de contraintes :

<Liste_contraintes> :: = {<Contrainte> }

Contrainte :

<Contrainte> :: = <Expression> <Comparateur> <Comparant> | <Sens_évolution> <Suite_contrainte>

Comparant :

<Comparant> :: = <Valeur> | <Expression>

Expression :

<Expression> :: = <Expression_simple> | <Borne> <Suite_contrainte>

Suite_contrainte :

<Suite_contrainte> :: = <Parenthèse_ouvrante> <Expression_simple> <Parenthèse_fermante>

Expression_simple :

<Expression_simple> :: = <Agrégat> | <Attribut>

Agrégat :

<Agrégat> :: = <Fonction_math> <Parenthèse_ouvrante> <Attribut> <Suite_attribut> <Parenthèse_fermante>

Attribut :

<Attribut> :: = <Dollar> <Chaîne> <Point> <Chaîne>

Attribut_entite :

<Attribut_entite> :: = <Lettre> <Attribut_entite>

Fonction_math :

<Fonction_math> :: = <Somme> | <Produit> | <Différence> | <Moyenne> | <Cardinalité> | <Distance>

Borne :

<Borne> :: = <Minimum> | <Maximum>

Sens_évolution :

<Sens_évolution> :: = <Minimisation> | <Maximisation> | <Maintient>

SuiteAttribut :

<Suite_attribut> :: = {<Virgule> <Attribut> }

Chaîne :

<Chaîne> :: = <Guillemet> (<Lettre>)+ <Guillemet>

Valeur :

<Valeur> :: = <Entier> | <Réel> | <Moins> <Valeur>

Réel :

<Réel> :: = <Entier> <Virgule> <Entier>

Règles de dérivation générant des tokens (terminaux) :

Label _ besoin :

<Label_besoin> :: = ‘Besoin’

Label _ scénario :

<Label_scénario> :: = ‘Scénario’

Deux_points :

<Deux_points> :: = ‘:’

Point_virgule :

<Point_virgule> :: = ‘;’

Entier :

<Entier> :: = ('0' .. '9')+

Evaluer :

<Evaluer> :: = ‘évaluer’

Classer :

<Classer> :: = ‘ordonner’

Tailler :

<Tailler> :: = ‘tailler’

Mise_à_jour :

<Mise_à_jour> :: = ‘MAJ’

Sélectionner :

<Sélectionner> :: = ‘sélectionner’

Champ_contrainte :

<Champ_contrainte > :: = ‘nom’ | ‘niveau_préférence’ | ‘complexité’ | ‘poids’

Somme :

<Somme> :: = ‘∑’

Produit :

<Produit> :: = ‘∏’

Différence :

<Différence>:: = ‘ différence’

Cardinalité :

<Cardinalité> :: = 'nombre'

Moyenne :

<Moyenne> :: = ‘moyenne’

Distance :

<Distance> :: = ‘distance’

Minimum :

<Minimum>:: = ‘min’

Maximum :

<Maximum> :: = ‘max’

Minimisation :

<Minimisation>:: = ‘minimiser’

Maximisation :

<Maximisation> :: = ‘maximiser’

Maintient :

<Maintient> :: = ‘maintenir ’

Dollar :

Dollar :: = ‘$’

Guillemet :

Guillemet :: = ‘"’

Lettre :

<Lettre> :: = ‘a’..‘z’ |‘A’..‘Z’ |‘0’..‘9’ |‘>’ |‘<’ |‘=’ |‘;’ |‘?’ |‘$’ |‘+’ |‘!’ |‘(’ |‘)’ |‘[’ |‘]’ |‘-’ |‘_’ |‘\’ |‘/’ |‘~’ |‘:’ |‘/’
|‘{’ | ‘}’ | ‘*’ |‘%’ |‘#’

Parenthèse-ouvrante :

<Parenthèse_ouvrante> :: = ‘(‘

Parenthèse-fermante :

<Parenthèse_fermante> :: = ‘)’

Virgule :

<Virgule> :: = ‘,’

Point :

<Point> :: = ‘.’

Comparateur :

<Comparateur> :: = ‘<’ | ‘>’ | ‘=’ | ‘<>’ | ’<=’ | ‘>=’ | ‘∈’

Annexe 3 :

Automate cellulaire de simulation du risque

La notion de base d'automate cellulaire repose sur une partition de l'espace d'étude en cellules régulières (le plus souvent carrées) reliées par leur topologie de voisinage de 4 ou 8 voisins. Chaque cellule est un automate à nombre d'états fini. Des règles[87] de transition appliquées aux cellules permettent de calculer leur état courant à partir de leur ancien état et de celui des cellules voisines, à l'instant précédent.

Le processus de simulation (figure 73) commence par sélectionner graphiquement (ou importer depuis un fichier) une portion du M.N.T qui sert à construire les cellules et le graphe d'écoulement de l'automate. L'interface de l'automate (figure 72) permet d'effectuer différentes visualisations sur le modèle de surface (pentes, expositions, courbes de niveaux, lignes d'écoulements, zones inondables), mesurer et délimiter des bassins versants, calculer des courbes de débits en différents points de mesure, etc. [LANGLOIS 02].

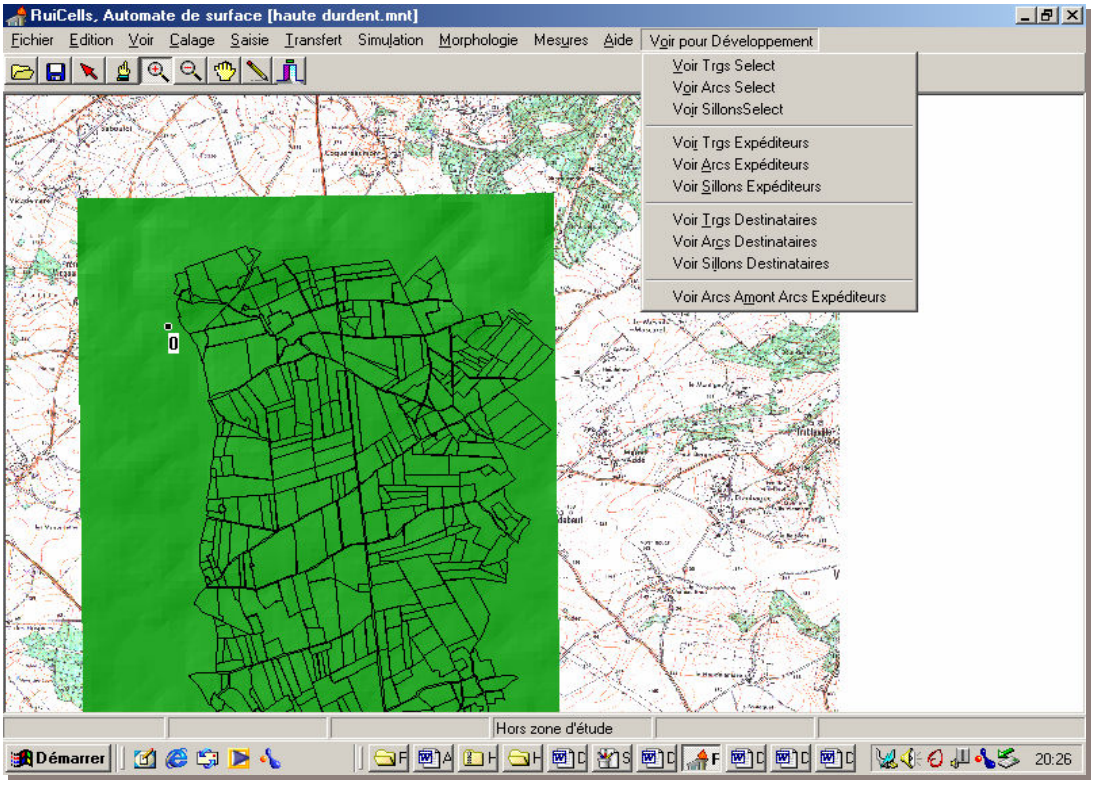


Figure 72 : Interface de l'automate cellulaire.

[87] Ces règles peuvent être déterministes, probabilistes, de type chaînes de Markov, etc.

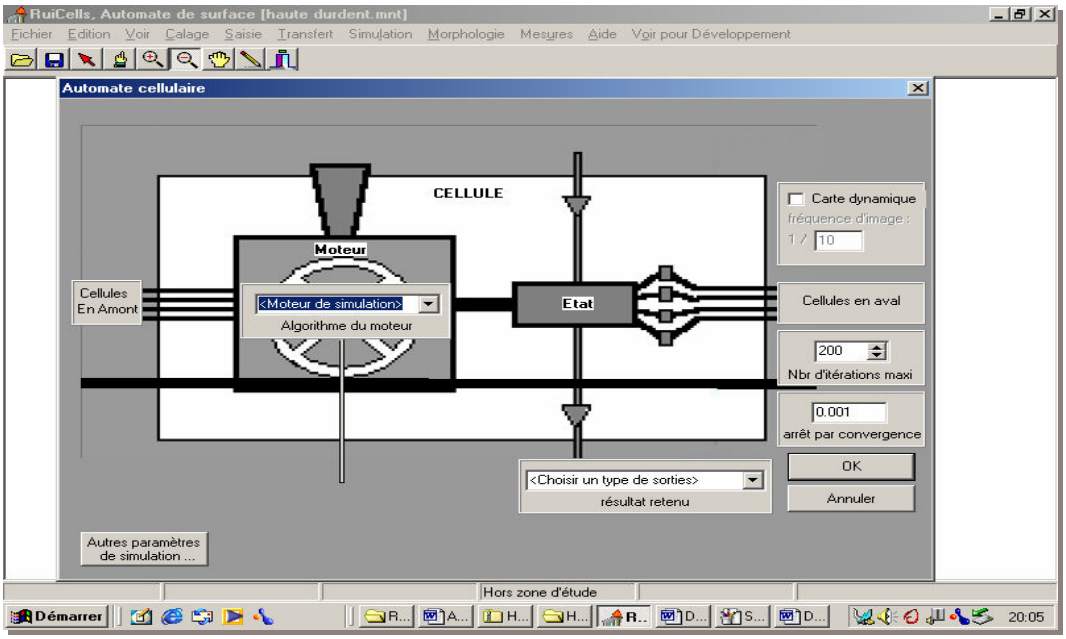


Figure 73 : Moteur de simulation de l'automate cellulaire.

Les cellules :

Les cellules de l'automate sont associées chacune à un élément unique du graphe topologique (triangle, arc ou pôle). Elles contiennent de nombreuses propriétés lui permettant de communiquer, de connaître la géométrie et la topologie de l'élément du graphe associé, mais aussi son contenu descriptif : type d'occupation du sol, coefficient de ruissellement, précipitation, quantité d'eau accumulée, etc.

En entrée, une cellule peut être reliée à un nombre quelconque de cellules amont, même si ce nombre dépasse rarement deux. Les liens sortants d'une cellule, de même, dépassent rarement deux (figure 74). Le flux sortant calculé sur la cellule est partagé entre ses différentes sorties selon les proportions de la surface qui s'écoulent sur un bord ou un autre.

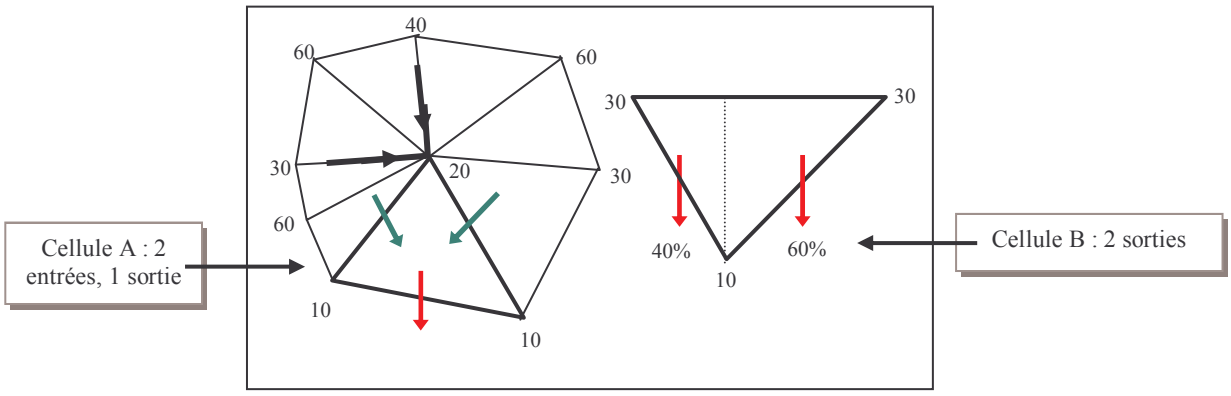


Figure 74 : Entrées et sorties d'une cellule.

La dynamique du système :

L'objectif de l'automate est de simuler les relations hydrologiques entre les cellules, puis de restituer les résultats sous une forme cartographique (figure 75) permettant de repérer les zones les plus sensibles au risque. Son fonctionnement global est itératif synchrone. Après une phase d'initialisation des cellules, la phase d'itération comporte une étape de communication entre ces cellules et une étape d'évaluation du nouvel état de chacune.

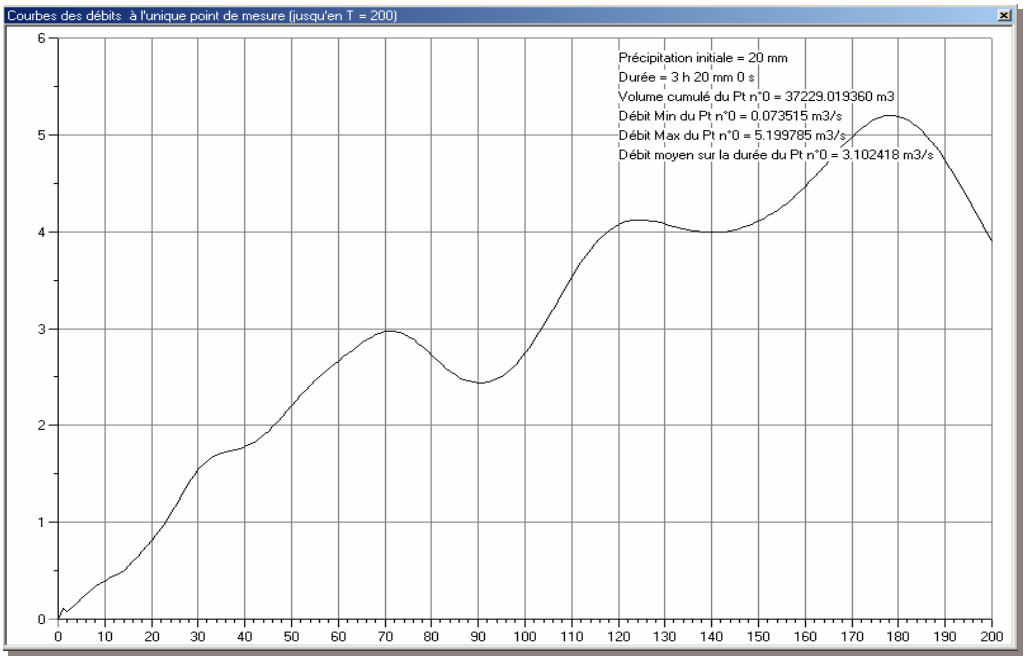


Figure 75 : Résultat fourni par l'automate, relatif à la courbe des débits.

Le fonctionnement du système dépend de la dynamique de chaque cellule, qui, en plus de ses propriétés descriptives, contient un « moteur » qui assure son évolution cellulaire (figure 76). De plus, le logiciel permet à l'utilisateur de choisir le type de simulation qu'il veut effectuer. En fonction de ce choix, des moteurs différents sont affectés aux cellules. Le processus peut prendre fin soit par suite de la convergence du système, aboutissant à un régime stationnaire qui est atteint lorsque l'état de toutes les cellules est égal à l'état précédent, soit par dépassement d'un nombre d'itérations fixé par l'utilisateur. Nous renvoyons à la lecture de [LANGLOIS 02] pour de plus amples détails sur l'automate cellulaire de simulation du risque de ruissellement.

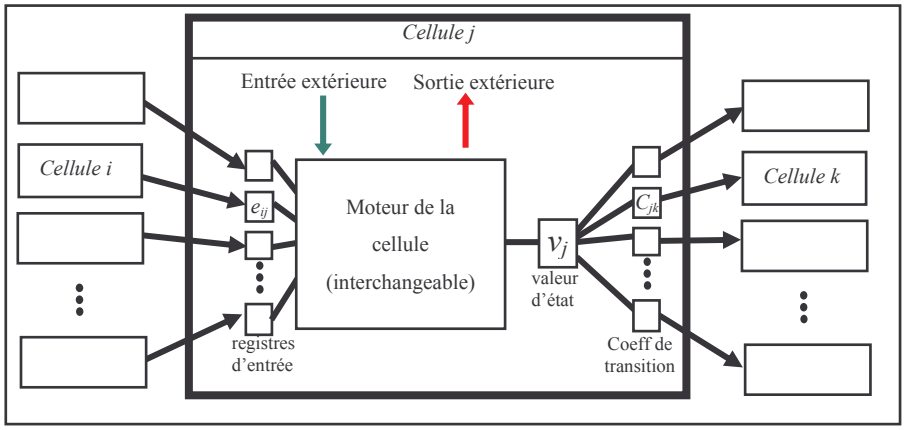


Figure 76 : Schéma de principe d'une cellule.

Annexe 4 :

Modèle multi-agents de résolution

A partir des valeurs spatialisées produites par l'automate cellulaire de simulation du risque, nous proposons une démarche d'optimisation des assolements visant à réduire les processus de risque tout en respectant un ensemble de contraintes agraires, économiques et réglementaires. Le problème est modélisé dans le cadre du paradigme Multi-Agents. A partir d'une occupation initiale relevée sur le terrain, l'optimisation du placement consiste à élaborer des simulations d'aménagement basées sur des redistributions spatiales des occupations du sol afin de réduire les risques tout en maintenant les objectifs de production des exploitants. Un groupe d'agents est associé à chaque type de culture. L'objectif de chaque groupe consiste à occuper une surface totale qui lui permette d'atteindre son objectif de production.

Environnement du système :

L'environnement de résolution du problème est constitué des différentes parcelles à occuper par un agent d'un groupe de culture. L'ensemble du territoire est considéré propriété d'un seul exploitant. Un groupe d'agents est associé à chaque type de culture. L'objectif de chaque groupe consiste à occuper les parcelles les moins sensibles du territoire tout en restant proche de son objectif de production. Les agents placés sur des parcelles très sensibles vont essayer de changer de localisation en attaquant les agents des autres groupes. Nous gardons le nombre initial d'agents pour chaque groupe et qui représente le nombre de parcelles occupées initialement par ce type de culture. Les agents sont placés initialement sur les parcelles du territoire.

Nous désignons par ***P*** l'ensemble de parcelles présentes dans le territoire. Chaque parcelle ***p*** est caractérisée par :

- Sa surface : **Surf(p)**.
- La valeur de sa pente : **Pente(p)**.
- Son type de sol : **TypeSol(p)**.

- Sa surface en amont : **SA(p)** qui correspond à la partie du territoire qui fait écouler l'eau vers cette parcelle.

- Une valeur de sensibilité : **SS(p)** qui représente la quantité d'eau qui arrive à cette parcelle.

- Son exploitation : **Exp(p)** qui représente la ferme à laquelle appartient cette parcelle.

- L'agent occupant : **a(p)** qui représente un type de culture.

Les agents :

Un agent est une entité appartenant à un groupe et ayant les mêmes objectifs et les mêmes caractéristiques que les autres agents de son groupe. L'ensemble des agents est noté ***A***. Un agent ***a*** est caractérisé par :

- Sa localisation : **p(a)** qui représente la parcelle occupée par cet agent à un instant t de la simulation.

- Son groupe de culture : **g(a)**.

- La quantité de risque local : **RL(a)** exprimée en fonction du coefficient de risque de son groupe et de la sensibilité de la parcelle qu'il occupe.

- Sa réalisation en production à l'objectif de son groupe : **R(a)= Surf(p(a))**. Elle est égale à la surface de la parcelle **p(a)** occupée par cet agent.

Les groupes d'agents :

Il y a autant de groupes d'agents que de types de cultures. L'ensemble des groupes est noté **G**. Une culture est représentée par un groupe **g** d'agents. Un groupe **g** est décrit par un ensemble de paramètres :

- Le type de sol compatible : **SolCompatible(g)**.

- La surface de parcelle favorable à sa plantation : **SurfFavorable(g)**.

- La valeur de pente compatible : **PenteCompatible(g)**.

- Un nombre d'agents : **Nba(g)** égal au nombre de parcelles occupées par la culture g.

- Un objectif à atteindre : **O(g)** qui s'exprime en terme de quantité à produire. Nous considérons que l'objectif d'un groupe est égal à sa réalisation à l'instant t = 0 de la résolution.

- Un coefficient de risque : **Coef(g)** qui représente l'effet de ce type de culture vis à vis de l'absorption de l'eau. Un coefficient négatif indique que la culture absorbe de l'eau et lutte donc contre le risque.

- Un seuil maximum de satisfaction : **SeuilMax(g)** exprimant le dépassement maximum de l'objectif de production autorisé pour le groupe de culture.

Evolution des groupes d'agents :

Nous introduisons trois grandeurs variables au cours du temps et qui définissent l'évolution des groupes d'agents :

- La Réalisation du groupe : $R(g) = \sum R(a)$ à un instant t, avec **a** un agent du groupe **g**. Elle est égale à la somme des réalisations de ses agents.

- La Satisfaction du groupe : $S(g) = R(a) / O(g)$. Une valeur égale à 1 correspond au cas où la réalisation du groupe est égale à son objectif final.

- La Distance à l'objectif : $Dist(g) = | 1 - S(g) |$. A l'initialisation du système, nous partons avec des distances nulles pour les différents groupes. Plus la réalisation du groupe s'approche de son objectif, plus sa distance vers cet objectif diminue.

Stratégie de résolution :

Nous partons d'une occupation initiale prélevée par les experts agronomes qui ont observé les répartitions d'occupations du sol réalisées par les agriculteurs chacun dans son exploitation.
Chaque parcelle est occupée par un agent d'un groupe. A chaque pas de la résolution, l'agent placé sur la parcelle la plus sensible du territoire, appelé agent agresseur, cherche une possibilité de permutation avec les agents des autres groupes. Les agents agressés doivent être localisés sur des parcelles moins sensibles que celle de l'agent agresseur. Ceci étant afin d'éviter de placer sur les parcelles les plus sensibles du bassin versant une culture plus ruisselante que celle qui y existait. Chaque permutation doit apporter un gain en terme de risque local généré par les deux agents concernés. Toutefois, l'agent agresseur ne s'autorise

pas à effectuer une permutation qui détériore fortement (qui dépasse le seuil de satisfaction autorisé) la satisfaction de son groupe. De la même manière, l'agent agressé refuse la permutation qui peut générer le dépassement de l'intervalle de satisfaction admissible pour l'objectif de son groupe ou qui ne respecte pas ses conditions agronomiques de localisation (surface, pente et type de sol). L'agent agresseur choisit parmi les agents agressés un agent candidat pour la permutation en fonction de la valeur du risque local et de la satisfaction des groupes des deux agents en question. A l'issue de cette étape, nous avons recours à l'évaluation de la valeur du risque global dans tout le territoire afin de décider de la validation ou de l'annulation de la permutation. Nous définissons la notion de jetons pour exprimer le nombre d'essais dont dispose un agent pour valider une permutation avec un agent candidat. Le même nombre de jetons est accordé à chaque agent durant une simulation. Chaque agent dispose donc d'un nombre déterminé de jetons lui permettant d'effectuer des essais consécutifs en cas de non-validation de sa permutation avec l'agent candidat. L'évolution du système peut donc avoir 3 directions :

- La valeur du risque global s'améliore et la permutation des deux agents est validée. Un autre agent occupant la parcelle sensible suivante est alors activé.

- La valeur du risque ne s'améliore pas et dans ce cas l'agent agresseur perd un jeton et choisit un autre agent candidat pour la permutation.

- L'agent agresseur a consommé tous ses jetons et termine donc sa session.

L'agent agresseur termine sa tâche par un succès en occupant une autre parcelle ou par un échec s'il ne trouve pas d'agents candidats à une permutation qui vérifie les conditions citées ci-dessus ou s'il ne dispose plus de jetons. A chaque instant, l'état de l'agent peut être actif ou endormi. Un agent est actif s'il est agresseur ou agressé. Il est endormi lorsqu'il n'est pas concerné par une permutation. Un agent endormi peut être réveillé par un autre agent qui le sélectionne pour une éventuelle possibilité de permutation. La condition d'arrêt du système est marquée par l'obtention d'un état stable caractérisé par l'impossibilité d'échanges de parcelles entre les agents. Le principe de l'algorithme est le suivant :

Début

1. Sélection de l'agent agresseur placé sur la parcelle la plus sensible.
2. Recherche de tous les agents des autres groupes qui occupent une parcelle moins sensible conforme aux exigences de l'agresseur (pente, sol et taille) et qui permettent l'amélioration du risque local.

3. Eliminer les agents dont la permutation avec l'agent agresseur entraîne le dépassement des seuils de satisfaction de l'un des groupes de cultures.
4. Choisir un agent candidat pour la permutation en fonction de la satisfaction des deux groupes d'agents mis en jeu et de la valeur du risque local.
5. Calculer le risque global à l'exutoire du bassin en cas de permutation avec l'agent candidat.
6. Effectuer la permutation ou reprendre l'étape 4 si l'agent agresseur dispose encore de jetons.
7. Aller en 1 tant qu'il existe un agent non encore activé.

Fin

Le système multi-agents a alors pour principal rôle de proposer à chaque itération une permutation de deux agents dans le but d'améliorer la valeur du risque dans le territoire. Chaque permutation doit naturellement respecter les conditions d'affectation locales tout en garantissant des objectifs de production des groupes dans la limite du fonctionnement admissible fixé par les seuils de satisfaction. Le modèle développé a été étendu pour prendre en compte la pluralité des exploitations et modéliser ainsi un type d'agents supplémentaire, relatif aux exploitants. Ce modèle ne sera pas présenté dans cette annexe pour des raisons de simplicité.

Résumé :

Les travaux présentés dans ce mémoire traitent du problème de la modélisation des contraintes et de l'interaction système-utilisateurs pour la résolution de problèmes complexes et sur-contraints. La satisfaction des contraintes des décideurs constitue un élément fondamental dans tout processus de décision, contraint la marge de manœuvre du système et la qualité de la réponse pouvant être apportée au problème étudié. Cependant, il semble que cet aspect soit peu défini, manque de modélisation et de fondements génériques de traitement et de gestion. L'intérêt reste souvent porté sur la démarche et les techniques de résolution plutôt que sur la modélisation du problème lui-même. Ce travail se focalise sur ce besoin de modélisation. Nous proposons une nouvelle démarche interactive de progression dans la résolution des problèmes d'optimisation sous contraintes, notamment en intégrant les décideurs dans la définition de leurs contraintes et dans la prise de décision finale afin de les aider à décider au lieu de décider à leur place. L'approche développée a été appliquée pour l'optimisation des assolements dans la prévention des risques de ruissellement touchant les territoires agricoles. Les simulations ont été effectuées sur le territoire du Pays de Caux au nord-ouest de la France.

Mots-clés : Modélisation des contraintes, optimisation, aide à la décision, risque de ruissellement.

Abstract:

Works presented in this thesis deal with the problem of constraints modelling and the system-users interaction to solve complex and over-constrained problems. The satisfaction of decision-maker's constraints constitutes a fundamental element in any decision process and constrains the margin of the system and the quality of solution that can be offered. However, it seems that this aspect received few propositions and lacks of modelling and generic treatment. The interest is often carried on the gait and the techniques of resolution rather than the modelling of the problem and it becomes necessary to pay more attention to fill this need of modelling. In this orientation, we propose a new interactive gait of progression in the resolution of the constraint satisfaction and optimisation problems, notably while integrating the decision-makers in the definition of their constraints and in making a final decision in order to help them to decide instead of deciding at their place. The developed approach was applied for the optimisation of crops assignment in order to minimize the streaming risks concerning agricultural territories. The results of simulation were obtained for the territory of Pays de Caux in the northwest of France.

Key words: Constraints modelling, optimisation, decision-aid, risk of streaming.