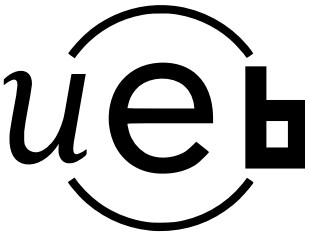
Analyse multidimensionnelle interactive de résultats de simulation. Aide à la décision dans le domaine de l'agroécologie

Tassadit Bouadi

N° d'ordre : 00000 ANNÉE 2013



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Tassadit BOUADI

préparée à l'unité de recherche IRISA – UMR6074
Institut de Recherche en Informatique et Systèmes Aléatoires
ISTIC

Analyse multidimensionnelle interactive de résultats de simulation. Aide à la décision dans le domaine de l'agroécologie.

Thèse à soutenir à Rennes le 28 novembre 2013

devant le jury composé de :

Ladjel BELLATRECHE
Professeur à l'Université de Poitiers, ENSMA / *Rapporteur*

Fadila BENTAYEB
Maître de conférence (HDR) à l'Université de Lyon 2 / *Rapporteur*

David GROSS-AMBLARD
Professeur à l'Université de Rennes 1 / *Examinateur*

Jean-Louis DROUET
Chargé de recherche INRA / *Examinateur*

Frédérick GARCIA
Directeur de recherche INRA / *Examinateur*

René QUINIOU
Chargé de recherche INRIA / *Examinateur*

Marie-Odile CORDIER
Professeur à l'Université de Rennes 1 / *Co-directrice de thèse*

Chantal GASCUEL-ODOUX
Directrice de recherche INRA / *Co-directrice de thèse*

J'adore parler de rien, c'est le seul domaine où j'ai de vagues connaissances
Oscar Wilde

Table des matières

Introduction

Contexte

La maîtrise des impacts des pratiques agricoles sur la qualité des eaux de surface est un enjeu important d'un point de vue environnemental (eutrophisation), économique et social (pêche, tourisme), et de la santé publique (eau potable, pêche,...). Dans le but d'aider à la gestion de l'eau et des activités agricoles, des modèles agro-hydrologiques ont été développés [CGG$^+$05, GOAC$^+$09b]. Ils couplent des modèles de cultures, représentant le développement des cultures, et des modèles de transfert de polluants, nitrates ou pesticides par exemple, dans les bassins versants agricoles à fortes contraintes environnementales. Le modèle TNT en est un exemple [BDR01, BDR$^+$02]. Il permet de simuler l'impact de scénarios de changements des pratiques agricoles sur les transferts et les transformations de l'azote dans un bassin versant. Ce modèle a été appliqué sur un des bassins versants qui alimente la baie de la Lieue de Grève près de Lannion, le bassin versant du Yar qui contribue aux trois quart des flux d'azote émis dans cette baie. En effet, cette baie souffre chroniquement de phénomènes d'eutrophisation qui se manifestent par le développement de macroalgues (marées vertes). Les concentrations en nitrates sur ce cour d'eau du Yar sont malgré tout modérées, de l'ordre de 30 mg/l. Les objectifs à atteindre en termes de concentrations sont ambitieux, de l'ordre de (10-15 mg/l). Les agro-systèmes doivent donc évoluer.

Cette thèse a été financée dans le cadre du projet ANR ACASSYA [1]. Ce projet s'attache à trois objectifs majeurs : (i) comprendre et évaluer les flux et les dynamiques du cycle de l'azote dans une bassin versant, (ii) modéliser les exploitations et leurs émissions au niveau du bassin versant, en intégrant les contraintes des systèmes d'élevage, les structures du milieu et des paysages, iii) faciliter la mise à disposition et l'exploration des données de simulations par les experts et les gestionnaires du bassin versant, et (iv) co-construire et mettre en place avec les acteurs des scénarios d'évolution des systèmes agricoles sur le territoire. Ma thèse s'inscrit dans l'axe 3. Son objectif a été de développer des méthodes d'aide à l'analyse des résultats de simulation obtenus par le modèle TNT. La finalité est d'obtenir un outil permettant à des acteurs, gestionnaires de bassins versants ou chargés d'étude, d'explorer des résultats de simulations pour

1. ACcompagner l'évolution Agro-écologique deS SYstèmes d'élevAge dans les bassins versants côtiers (ANR-08-STRA-01).

élaborer des solutions efficaces aux problèmes posés. L'idée de base est de s'appuyer sur la demande de l'utilisateur pour analyser les résultats de simulation et, de manière interactive, extraire des connaissances l'aidant à mieux comprendre une situation et à mieux évaluer l'impact d'une décision. Répondre à une question de l'utilisateur, c'est répondre à une requête sur une base de données rassemblant l'ensemble des résultats de simulations, puis fournir des moyens d'analyse, d'exploration et de visualisation de la réponse.

Analyser les résultats de simulations est difficile en raison du grand nombre de données (entrées / sorties, données intermédiaires, ...). Pourtant, l'analyse de ces résultats est indispensable dans un processus d'aide à la prise de décision. Parmi les démarches proposées pour l'aide à la décision, nous trouvons la simulation de scénarios (ex. les travaux menés au sein de l'équipe DREAM [LCB+11, ZLC11]), qui expriment des questions du type *"What If"* (en français *"Que se passerait-il si"*) ou plus précisément comment évoluent les sorties quand les entrées varient, et l'apprentissage symbolique [TSAC+05, TSAC+13, GOAC+09a] permettant, à partir de résultats de simulation, d'induire des règles décrivant l'impact, c'est-à-dire d'identifier des facteurs pertinents utiles pour la prise de décision et la recommandation d'actions. Cependant, on peut regretter l'absence de l'utilisateur dans ces démarches alors même que le processus d'aide à la prise de décision est censé être centré utilisateur. Les motivations de cette thèse consistent à élaborer des méthodes d'analyse des résultats de simulation qui replacent l'utilisateur au coeur du processus décisionnel, et qui permettent d'analyser et d'interpréter de gros volumes de données de manière efficace. L'objectif est de mettre ces données de simulation dans les mains des acteurs, pour qu'ils puissent se les approprier.

La démarche développée est d'utiliser des méthodes d'analyse multidimensionnelle interactive. Peu de travaux traitent de cette question dans les domaines de l'agronomie et de l'environnement. Nous proposons tout d'abord une méthode d'archivage des résultats de simulation dans une base de données décisionnelle (i.e. entrepôt de données), adaptée au caractère spatio-temporel des données de simulation produites. Nous proposons ensuite d'analyser ces données de simulations avec des méthodes d'analyse en ligne (OLAP) afin de fournir aux acteurs des informations stratégiques pour améliorer le processus d'aide à la prise de décision. Enfin, nous suggérons de coupler l'analyse en ligne avec les requêtes skyline afin de permettre aux acteurs de formuler de nouvelles questions en combinant des critères environnementaux contradictoires, et de trouver les solutions compromis associées à leurs attentes, puis d'exploiter les préférences des acteurs pour détecter et faire ressortir les données susceptibles de les intéresser.

Contributions

Dans ce travail de thèse nous proposons une méthodologie de construction d'un entrepôt de données qui stocke les données de simulation d'un modèle agro-hydrologique, dont les principales étapes sont : (i) le pré-traitement des données de simulation, (ii) la

modélisation multidimensionnelle et hiérarchique des pratiques agricoles, et (iii) l'analyse des résultats de simulation en combinant la modélisation spatio-temporelle des données, l'entreposage des données et l'analyse en ligne. Cette méthodologie peut être appliquée à une variété de problématiques agro-environnementales, notamment, pour l'analyse des interactions entre pratiques agricoles et milieu, et la synthèse d'informations environnementales. Ce modèle d'entrepôt de données a été couplé avec un système d'information géographique (SIG) afin de permettre une visualisation cartographique des résultats de simulation stockés, et de faciliter le processus d'analyse et d'aide à la décision.

Dans ce travail, nous manipulons des données volumineuses, ce qui engendre un entrepôt de données volumineux et dense. Par ailleurs, l'analyse en ligne classique ne met pas à disposition des utilisateurs des d'outils permettant de le guider vers les données les plus intéressantes de l'entrepôt. C'est à l'utilisateur de manipuler au mieux l'entrepôt de données pour y découvrir les zones d'information pertinentes pour les questions qu'il se pose. Dans le contexte de données volumineuses, une telle navigation peut s'avérer fastidieuse, voire rédhibitoire. C'est ce qui justifie le recours à des méthodes spécifiques pour détecter automatiquement les données susceptibles d'intéresser l'utilisateur. Nous nous sommes ainsi intéressés aux requêtes skyline. En effet, ces requêtes sont des requêtes multi-critères avec préférences. Elles permettent de sélectionner à partir d'une base de données volumineuse, les "meilleurs" tuples selon des préférences exprimées par l'utilisateur. Ce type de requêtes répond complètement aux besoins que nous avons exprimés (i.e. prise en compte de l'utilisateur dans le processus décisionnel). Nous proposons donc de coupler les requêtes skyline avec l'analyse en ligne.

Deux méthodes d'extraction de skyline ont été développées dans le contexte des entrepôts de données.

La première méthode, EC^2Sky, se focalise sur comment répondre efficacement à des requêtes de type skyline en présence de préférences utilisateurs dynamiques, et ce malgré de gros volumes de données. Les préférences dynamiques sont des préférences qui changent d'un utilisateur à un autre. Ce problème constitue un véritable challenge dans le contexte des bases de données volumineuses (ex : les entrepôts de données). Parmi les rares auteurs ayant abordé ce problème, Wong et ses collègues [WFP+08] ont récemment introduit le concept de *préférence d'ordre n*, et la propriété (nommée *merging property*) permettant de traiter les raffinements des préférences sur une seule dimension. Ils ont démontré que le skyline associé à n'importe quelle préférence sur une dimension donnée pouvait être calculé à partir des préférences de premier ordre de cette même dimension. Ils proposent une méthode de semi-matérialisation ($IPO\text{-}Tree$) basée sur une structure de données qui stocke les résultats partiels correspondant à toutes les combinaisons possibles des préférences de premier ordre des différentes dimensions dynamiques. Par conséquent, la taille de la structure induite par cette matérialisation est de l'ordre de $O(c^m)$, où m représente le nombre de dimensions associées à des préférences dynamiques, et c la cardinalité maximale d'une dimension.

Cette méthode ne résiste donc pas au passage à l'échelle, et devient vite inexploitable

dans un contexte de données volumineuses. De plus, il est intéressant de noter que la *merging property* s'applique à une seule dimension à la fois.

Tout en réutilisant la *merging property* proposée par Wong et al., nous proposons une méthode incrémentale, appelée EC^2Sky, pour le calcul des skyline associés à plusieurs dimensions dynamiques. En effet, le skyline peut changer lorsque les préférences sont mises à jour, et devrait donc être progressivement maintenu pour éviter une réévaluation globale du skyline.

L'idée principale de EC^2Sky repose sur l'ajout incrémental des dimensions dynamiques lors du calcul des skyline. L'avantage de cette proposition est double. D'une part, la complexité en terme de stockage mémoire des informations pré-calculées est réduite à $O(m * c)$. D'autre part, le nombre de tests de dominance diminue de manière significative. Contrairement à la méthode *IPO-tree*, EC^2Sky fournit à l'utilisateur un moyen d'exprimer des préférences et de les modifier en ligne sans être pénalisé par des temps d'attente trop longs. Les performances sont obtenues en ne stockant que le minimum d'informations requises afin d'autoriser des mises à jour simples et rapides.

Lorsque les dimensions sont hiérarchiques, il est pertinent pour un décideur de pouvoir naviguer le long des axes des dimensions hiérarchiques (i.e. spécialisation / généralisation) tout en assurant un calcul en ligne des points skyline correspondants. Cependant, il n'existe pas, à notre connaissance, de travaux conciliant extraction de points skyline et prise en compte de dimensions hiérarchiques. Par exemple dans le contexte des entrepôts de données, des hiérarchies peuvent être définies pour chaque dimension. Que deviennent alors les points skyline ? Doit-on recalculer tous les skyline pour chaque niveau hiérarchique ? Peut-on les dériver à partir des skyline de niveau supérieur ou inférieur ? Peut-on étendre les algorithmes existants de calculs de skyline sur des dimensions hiérarchiques ?

Une solution naïve serait, soit de recalculer à chaque niveau hiérarchique l'ensemble des skyline, soit de stocker pour chaque niveau hiérarchique l'ensemble des skyline associés. Cependant, ces solutions sont très coûteuses en termes de temps de calcul et/ou de stockage mémoire dans un contexte de données volumineuses. Nous proposons donc une seconde méthode, $HSky$, permettant de réduire l'espace mémoire requis tout en assurant un calcul efficace et interactif des points skyline à tous les niveaux hiérarchiques des différentes dimensions. Nous mettons en évidence un ensemble de propriétés permettant la formalisation des relations hiérarchiques entre les préférences associées aux différentes dimensions. Ces propriétés permettent à l'utilisateur de spécialiser (*drill-down*) ou de généraliser (*roll-up*) les valeurs sur lesquelles portent ces préférences, et de dériver les points skyline associés de manière efficace.

Ces contributions ont été motivées et expérimentées par l'application de gestion des pratiques agricoles pour l'amélioration de la qualité des eaux des bassins versants agricoles. En effet, nous proposons de coupler le modèle d'entrepôt de données agro-hydrologiques construit avec les méthodes EC^2Sky et $HSky$ afin de permettre aux utilisateurs d'analyser uniquement les données qui les intéressent dans l'entrepôt de données, en prenant en compte leurs préférences.

Organisation du mémoire

Ce mémoire de thèse est organisé en quatre parties. La première partie est dédiée à un état de l'art centré sur les méthodes d'analyse des bases de données multidimensionnelles. Dans le premier chapitre de cette partie, après avoir introduit les concepts de base relatifs aux entrepôts de données et à l'analyse en ligne, nous présentons brièvement les différents domaines d'application des entrepôts de données. Une attention particulière est portée au domaine des sciences agronomiques et environnementales. Ensuite, nous présentons un état de l'art synthétique des différents travaux menés dans le cadre de l'enrichissement de l'analyse OLAP avec des méthodes de fouille et de recherche d'information. Le chapitre 2 décrit la problématique de la recherche des skyline, les concepts associés, ainsi que les différents algorithmes permettant leur calcul dans le contexte des bases de données. Nous concluons ce chapitre par une discussion.

La deuxième partie de ce mémoire est dédiée aux contributions dans le cadre des entrepôts de données agro-environnementales. Dans le chapitre 3, nous présentons un modèle d'entrepôt de données agro-hydrologiques, *N-Catch* (*Nitrogen in Catchment data warehouse*), développé pour stocker et analyser des données issues de simulations du modèle *TNT*. Nous décrivons la méthodologie suivie pour la conception et la construction de *N-Catch*. Nous présentons les spécificités de *N-Catch* et l'efficacité de son processus de stockage et d'analyse des données de simulation pour l'aide à la décision. De plus, nous présentons le couplage de l'entrepôt de données *N-Catch* avec le système d'information géographiques *QGIS* afin de permettre une visualisation cartographique des résultats de simulation stockés, et de faciliter le processus d'aide à la décision.

La troisième partie de cette thèse est dédiée à nos contributions concernant le calcul des requêtes skyline en présence de préférences dynamiques ou hiérarchiques. Nous développons dans le chapitre 4 les aspects formels ainsi que l'implémentation de notre approche EC^2Sky. Cette approche permet un calcul incrémental des skyline en présence de préférences dynamiques. Des expérimentations conséquentes, réalisées sur des données synthétiques, soulignent la pertinence de la solution proposée en la comparant aux références du domaine. Dans le chapitre 5, nous détaillons les aspects formels ainsi que l'implémentation de notre proposition *HSky* qui étend la recherche des points skyline aux dimensions hiérarchiques. Là encore, nous avons mené des expérimentation conséquentes sur des données synthétiques qui confirment l'efficacité et la pertinence de la solution proposée.

La quatrième et dernière partie de ce mémoire est consacrée au couplage de l'entrepôt de données *N-Catch* avec les algorithmes EC^2Sky et *HSky*. Afin de démontrer l'utilité de nos méthodes, nous appliquons nos algorithmes sur les données de simulations issues de *TNT*, dans le cadre de l'analyse multidimensionnelle des données stockées dans l'entrepôt de données *N-Catch*. Ce travail fait l'objet du chapitre 6.

Pour conclure ce mémoire de thèse, nous établissons un bilan des contributions apportées et nous traçons différentes perspectives de recherche qui pourraient être menées ultérieurement, tant du point de vue théorique, que du cas traité.

Première partie

État de l'art : Analyse des bases de données multidimensionnelles

Chapitre 1

Analyse en ligne OLAP

1.1 Introduction

L'évolution permanente des technologies de l'information conduit de plus en plus d'acteurs (entreprises, recherche,...) à conserver leurs données et ainsi préserver la mémoire de leurs activités. Les données collectées par ces acteurs sont un atout puissant pour dégager des tendances passées, actuelles et surtout futures. À partir des gisements de données ainsi constitués, il est naturel de chercher à les exploiter au mieux. Apparus pour gérer de très gros volumes de données issues de sources hétérogènes, les entrepôt de données [Inm92], ou en anglais *Data warehouses*, constituent l'outil essentiel de collecte et de mise à disposition des données en vue de leur analyse. L'analyse de ces données fait appel à des traitement OLAP (*On-Line Analytical Processing*), introduits par les auteurs de [CCS93], qui se distinguent des processus OLTP (*On-Line Transactional Processing*) principalement par leur complexité et par le nombre de données. En effet, il ne s'agit pas de formuler des requêtes classiques, simples et fréquentes, sélectionnant généralement quelques dizaines de tuples (i.e. enregistrements), mais de procéder à des analyses nécessitant d'agréger, de visualiser et d'explorer de manière interactive les données. On parle de navigation dans les données et d'analyse exploratoire. Pour cela les données sont représentées dans une structure particulière appelée cube de données ou hypercube [GBLP96]. C'est le concept central pour l'analyse OLAP. Le cube de données est constitué de l'union des résultats des requêtes agrégatives *Group-By* sur toutes les combinaisons possibles des critères d'analyse (ce que l'on nommera par la suite : *dimensions*). Grâce au pré-calcul du cube, l'utilisateur peut avoir une réponse quasi instantanée à toutes les requêtes qui lui seront utiles. Pour ce faire, le modèle d'analyse OLAP fournit des opérateurs pour résumer les données sous forme d'agrégats (ou au contraire pour détailler les éléments agrégés) et d'opérateurs pour visualiser les informations contenues dans le cube de données. Ces opérateurs de navigation sont généralement décomposés en trois catégories : opérateurs de structuration (*Rotate, Switch, Push, Pull*), de sélection (*Slice, Dice*) et d'agrégation (*Roll-up, Drill-down*). Dans le contexte de notre travail, nous nous intéressons particulièrement aux deux dernières catégories d'opérateurs. Cependant, à l'inverse de la fouille de données, OLAP ne permet pas d'extraire auto-

matiquement des connaissance (implicites ou explicites) à partir des données [Cha98]. En effet, une des limites de l'OLAP est de se restreindre à des aspects exploratoires et navigationnels. La fouille de données quant à elle, permet d'extraire des connaissances à partir des données et a une grande variété de méthodes avec des objectifs d'analyse différents. Ainsi une nouvelle problématique OLAP a fait son apparition. Dès la fin des années 90, plusieurs travaux [Cha98, Han97, SS01, Sar99, SAM98, GC01, CZC01] proposent d'associer les principes de l'OLAP aux méthodes de fouille de données pour enrichir l'analyse en ligne et ne plus la limiter à une simple exploration ou à une simple visualisation de données. Le couplage entre l'analyse en ligne et la fouille de données est alors désigné par les termes de *OLAM* (*On-Line Analytical Mining*) [Han97], *OLAP Intelligence*, *Multidimensional Mining*,...

Parallèlement à ces travaux, d'autres équipes de recherche se sont intéressées à l'association de l'OLAP avec des méthodes de recherche d'information. Les premiers travaux remontent au début des années 2000 [MLC$^+$00, PP03] et se sont particulièrement focalisés sur la combinaison de l'OLAP avec des techniques d'analyse de documents dans le cadre d'entrepôts de données textuelles. En effet, avec l'avènement des données complexes (données multi-format et/ou multi-structure et/ou multi-source, ...), l'analyse en ligne doit s'adapter à la nature spécifique de ces données tout en gardant l'esprit de l'OLAP. Ces travaux ouvrent de nouvelles pistes de recherche dans le contexte de l'enrichissement des possibilités de l'analyse OLAP et de l'intégration des connaissances de l'utilisateur dans le processus d'analyse. Nous portons une attention particulière à ces derniers travaux, car dans le contexte de notre étude, nous nous orientons vers une combinaison des principes de l'OLAP et de la recherche d'information et en particulier, de la recherche d'information multi-critères (requêtes skyline), permettant la visualisation et l'extraction de régions intéressantes dans un cube de données. Nous souhaitons, à travers cette approche, guider l'utilisateur vers les informations qui sont susceptibles de l'intéresser et lui permettre d'évaluer la pertinence de ces dernières afin de savoir si elles constituent ou non de nouvelles connaissances.

Dans ce chapitre, nous introduisons les concepts de base relatifs aux entrepôts de données et à l'analyse en ligne. Par la suite, nous présentons un état de l'art synthétique des différents travaux menés dans le cadre de l'enrichissement de l'analyse OLAP avec des méthodes de fouille de données et de recherche d'information.

1.2 Entrepôts de données : concepts et définitions

Au cours de ces dernières années, les entrepôts de données ont joué un rôle essentiel dans le domaine de l'informatique décisionnelle en soutenant et en améliorant les processus décisionnels des organisations. "Un entrepôt de données est une collection de données thématiques, intégrées, non volatiles, historisées et exclusivement destinées aux processus d'aide à la décision" [Inm92]. L'évolution des technologies a conduit à conserver les données pour assurer le suivi des activités. L'intérêt de l'utilisation d'un entrepôt de données est : (i) de fournir un accès facile et rapide à ce gros volume de données accumulées au fil du temps à partir de diverses sources de données et dans

divers formats (bases de données traditionnelles, fichiers xml, fichiers excel,..etc.) et (ii) d'analyser ces données pour prendre des décisions stratégiques et tactiques. Leurs utilisateurs, des décideurs, sont donc peu nombreux et s'intéressent non pas au détail des données mais à des tendances générales, selon tel ou tel critère.

1.2.1 Modèle multidimensionnel

La modélisation multidimensionnelle est la base des entrepôts de données et de l'analyse multidimensionnelle. Une *dimension* est un axe d'analyse du sujet étudié. Un modèle multidimensionnel fournit un support pour une analyse reposant sur plusieurs dimensions. Les données sont organisées de sorte à mettre en valeur le sujet étudié (i.e. analysé) et les différents axes d'analyse. Ces sujets d'analyses, nommés *faits*, peuvent représenter par exemple : *les ventes d'un produit*, *la quantité de pesticides ou d'azote appliquée par les agriculteurs*. Un *fait* consiste en un ensemble de *mesures* correspondant à des informations sur le sujet analysé. Dans la suite de ce chapitre, tous les exemples et illustrations porteront sur le domaine d'application de l'agro-hydrologie, et plus précisément sur l'analyse de l'impact des pratiques agricoles sur la pollution nitrique.

Exemple 1 *Considérant le modèle multidimensionnel décrit par la Figure.1.1. Le fait Rendement Agricole est représentée par trois dimensions : culture (ici Id-Cult), date (ici Id-Dat) et localisation (ici Id-Loc), et une mesure : rendement de la culture (kg/ha) (ici Rendement-Culture).*

1.2.2 Modèle multidimensionnel hiérarchique

L'objectif d'un entrepôt de données est de permettre aux utilisateurs de formuler des requêtes complexes et d'effectuer des analyses sur des données agrégées afin d'en dégager des propriétés implicites. Chaque dimension peut être associée à une ou plusieurs *hiérarchies* utilisées pour afficher les données multidimensionnelles à plusieurs niveaux de granularité. Les valeurs des mesures associées à un niveau de granularité plus grossier sont obtenues en synthétisant des valeurs de mesures de plus bas niveau. On dit alors que les valeurs sont agrégées. Une fonction *d'agrégation* (e.x. somme, moyenne, maximum, etc.) est associée à chaque mesure définie dans le modèle multidimensionnel hiérarchique.

Exemple 2 *Nous décrivons dans la Figure.1.2 deux différentes hiérarchies possibles sur la dimension Localisation. La première décrit une hiérarchie stricte (basée sur un ordre total) : chaque parcelle appartient à un bassin versant, qui à son tour appartient à une région, elle même située dans un pays, et la deuxième décrit une hiérarchie non-stricte (basée sur un ordre partiel) : chaque maille appartient à une parcelle et à un bassin versant (il n'existe pas de relation hiérarchique entre parcelle et bassin versant, car une parcelle peut appartenir à plusieurs bassins versants en même temps), qui à leur tour appartiennent à une région.*

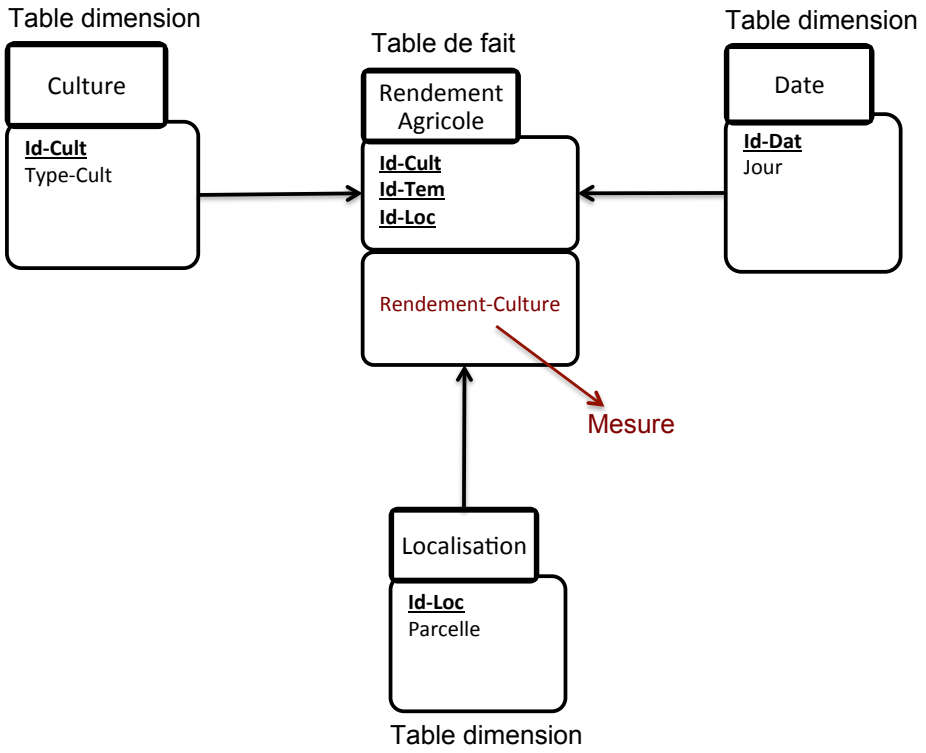


FIGURE 1.1 – Un exemple de modèle multidimensionnel avec trois dimensions (culture, date, localisation) et une mesure (rendement de la culture).

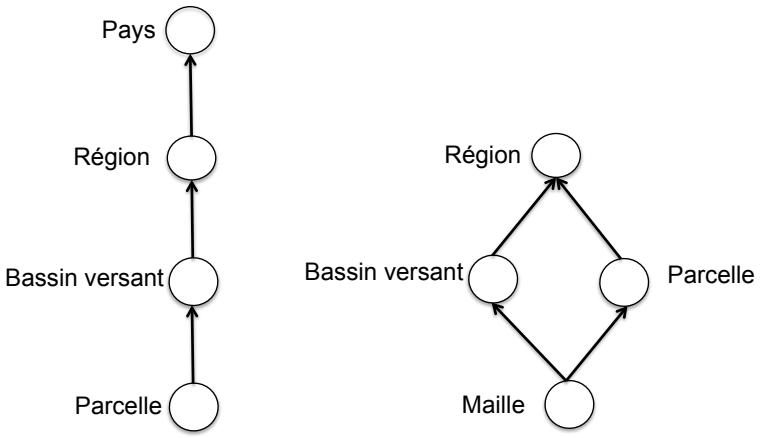


FIGURE 1.2 – Exemples de hiérarchies sur la dimension Localisation

Lors de l'analyse multidimensionnelle des données, les utilisateurs exploitent l'entrepôt de données en combinant les différentes dimensions et les différents niveaux de granularité des hiérarchies correspondantes. Pour sélectionner les données appropriées

au niveau d'abstraction adéquat, les utilisateurs expriment et soumettent des requêtes à l'entrepôt de données.

De telles requêtes sont particulièrement coûteuses car elles demandent le balayage d'important volumes de données. Cependant, ces requêtes s'inscrivent dans un processus d'aide à la décision et idéalement devraient être interactives. Pour concilier ces deux besoins contradictoires, i.e. répondre *rapidement* à des requêtes portant sur des données *volumineuses*, Gray et al. [GBLP96] ont introduit le concept de cube de données. C'est un concept central de l'analyse OLAP, pour pré-calculer et matérialiser tous les agrégats possibles. Ainsi, répondre à toute requête se réduit à une simple sélection de résultats préalablement stockés.

Un *cube de données* (ou cube OLAP) est une abstraction des données permettant aux utilisateurs de visualiser des données agrégées selon un ensemble de dimensions hiérarchiques. Les cellules du cube de données contiennent les valeurs des mesures associées aux dimensions et aux niveaux de granularité sélectionnés.

Exemple 3 *La Figure.1.3 décrit un cube de données représentant les rendements par culture, par année et par pays. Les agrégations sont calculées avec la fonction "somme".*

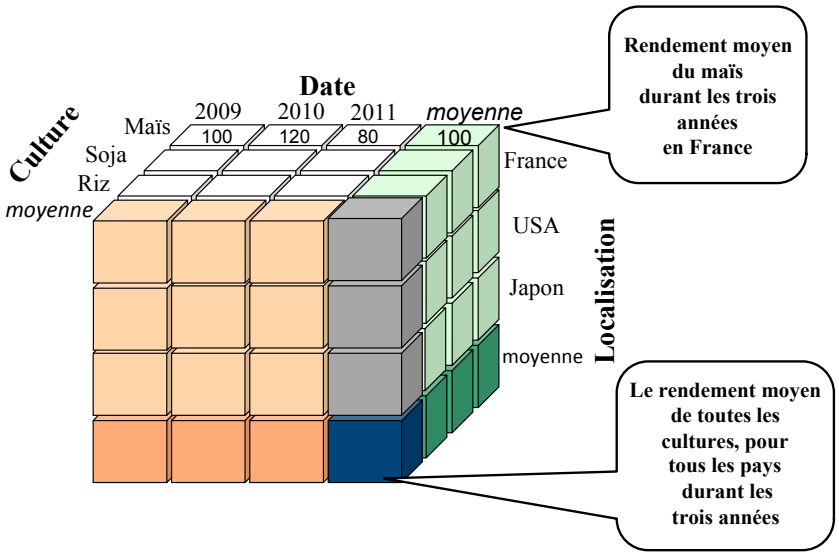


FIGURE 1.3 – Exemple de cube de données "Rendement Agricole"

1.2.3 Architecture d'un entrepôt de données

L'architecture d'un entrepôt de données comporte trois niveaux : les données constituent le premier niveau, le service du deuxième niveau est instauré par un serveur OLAP et les clients sont mis en oeuvre au dernier niveau, comme illustré par la Figure. 1.4. Les entrepôts de données sont alimentés par des sources de données externes grâce à des outils spécifiques appelés *ETL* (i.e., *Extract, Transform and Load*), en français *ETC*

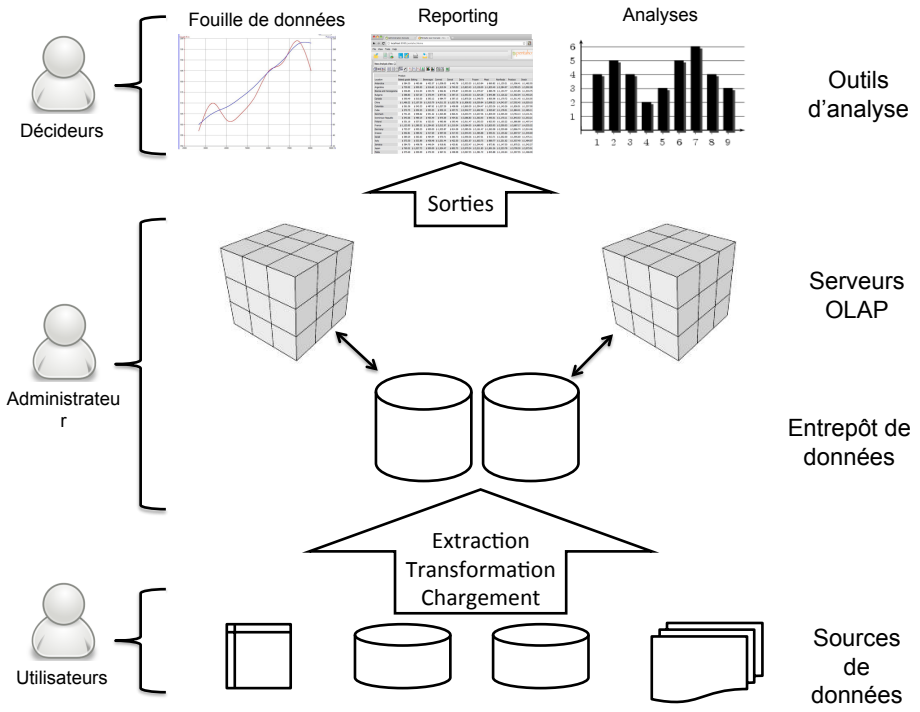


FIGURE 1.4 – Architecture d'un entrepôt de données

(*Extraction, Transformation et Chargement*). Ces outils extraient et pré-traitent des données provenant de sources hétérogènes et les chargent dans l'entrepôt de données.

Les serveurs OLAP créent les cubes OLAP à partir des données chargées dans l'entrepôt de données. Les utilisateurs accèdent et manipulent les cubes OLAP de manière optimale et facile grâce à un ensemble d'outils d'analyse. Ces outils offrent ainsi la capacité de réaliser à la volée des tableaux de synthèse, des rapports graphiques et des indicateurs pour réaliser des tableaux de bord. De plus, des outils de fouille de données sont parfois disponibles pour effectuer des analyses de tendances ou des prévisions.

1.2.4 Implémentations

Une fois le concept d'OLAP et ses propriétés définies, plusieurs implémentations ont été réalisées. Ces implémentations reprennent toutes les concepts OLAP, et chacune est optimisée pour une utilisation particulière d'OLAP. Dans la littérature, ces différentes implémentations peuvent être regroupées en plusieurs catégories : R-OLAP (Relational OLAP), M-OLAP (Multidimensionnal OLAP), H-OLAP (Hybrid OLAP), S-OLAP (Spatial OLAP), W-OLAP (Web-based OLAP) et RT-OLAP (Real-Time OLAP). Dans ce qui suit, nous nous focaliserons sur les quatre implémentations les plus utilisées et les plus intéressantes pour nos travaux : R-OLAP, M-OLAP, H-OLAP et S-OLAP (la partie sur S-OLAP sera développée ultérieurement dans le chapitre 3, section 3.5).

1.2.4.1 R-OLAP : Relational OLAP

L'implémentation R-OLAP ("Relational OLAP" ou OLAP relationnelle) utilise une base de données relationnelle, pour le stockage des données et des agrégations destinées à l'analyse. La majorité des fonctions nécessaires à l'utilisation d'OLAP sont donc gérées par un système de gestion de bases de données relationnelle (SGBDR). Le langage de requête (i.e. SQL) est expressif, et son interprétation est optimisée pour un temps de réponse minimal. La base de données relationnelle doit être structurée suivant un schéma spécifique afin de faciliter les processus d'analyse par le SGBDR. En effet, seuls certains schémas, décrits par les auteurs de [IBM98], peuvent être adaptés aux concepts OLAP. Le schéma en étoile est le plus connu, il consiste en une table de fait centrale et un ensemble de tables de dimensions représentées visuellement par une étoile (e.g. 1.1). En revanche, le schéma en constellation fusionne plusieurs schémas en étoile en utilisant des dimensions communes, il comprend donc plusieurs tables de faits et des dimensions communes ou spécifiques. Enfin, le schéma en flocon dérive du schéma en étoile : la table de fait est maintenue, tandis que les dimensions sont divisés en plusieurs tables en fonction de leurs hiérarchies. Le modèle en flocon est préconisé lorsque les tables de dimensions contiennent un très grand volume de données. Pour conclure, l'utilisation des SGBDR permet à l'implémentation R-OLAP de traiter un volume considérable de données. Cependant, l'implémentation R-OLAP se retrouve limitée aux fonctionnalités du langage SQL, et peut présenter des temps de calculs assez longs, étant donné qu'elle est plus adaptée à des processus transactionnels.

1.2.4.2 M-OLAP : Multidimensionnal OLAP

L'implémentation M-OLAP ("Multidimensional OLAP" ou OLAP multidimensionnel) utilise une base de données multidimensionnelle pour le stockage des données. Cette base de données multidimensionnelle possède une structure optimisée pour l'analyse, souvent nommée cube ou hypercube. M-OLAP utilise son propre système de gestion de bases de données, sous forme de matrices (i.e. tableaux). Toutes les fonctions spécifiques à l'OLAP sont mises en oeuvre nativement dans le système M-OLAP. Cette implémentation est généralement plus performante que R-OLAP pour plusieurs raisons. Tout d'abord, l'accès aux données par l'indexation multidimensionnelle est très peu coûteux. En effet, les données sont organisées de manière à être directement exploitables par les outils d'analyse. Ensuite, les techniques de compression permettent de réduire au minimum l'espace de stockage additionnel requis. Ce qui permet à l'implémentation M-OLAP d'occuper moins de place sur le disque que l'implémentation R-OLAP pour laquelle les agrégations sont stockées dans la base de données relationnelle. Cependant, M-OLAP présente des limites majeures sur le plan du passage à l'échelle. En effet, l'implémentation M-OLAP est très performante pour les données à faible dimensionnalité. Toutefois, en présence de gros volumes de données à forte dimensionnalité, la maintenance de la structure et les requêtes d'analyse deviennent trop coûteuses. C'est de ce constat qu'à émergé une nouvelle implémentation combinant R-OLAP et M-OLAP, appelé Hybrid OLAP.

1.2.4.3 H-OLAP : Hybrid OLAP

L'implémentation H-OLAP ("Hybrid OLAP" ou OLAP hybride) constitue un croisement entre les deux implémentations présentées précédemment. Au sein de cette implémentation, un serveur H-OLAP accède à deux bases de données différentes. La première, multidimensionnelle, contient les données agrégées. La seconde, relationnelle, contient les données détaillées. L'implémentation H-OLAP hérite des caractéristiques des implémentations R-OLAP et M-OLAP selon le type de données manipulées par les requêtes, R-OLAP dans le cas de données détaillées et M-OLAP dans le cas de données agrégées. Cette implémentation devient particulièrement intéressante lorsque la majeure partie des requêtes manipulent des données agrégées (l'accès au données dans M-OLAP est peu coûteux) et que les données détaillées représentent un volume important (les SGBDR permettent de traiter un volume considérable de données).

1.2.5 Opérateurs de navigation

La vocation de l'OLAP est de réaliser une analyse interactive et multidimensionnelle des données de l'entrepôt de données. On parle de navigation dans les données et d'analyse exploratoire. Cette analyse en ligne agrège les données pour pouvoir les explorer et les visualiser. L'OLAP dispose d'un ensemble d'opérateurs de navigation qui vont lui permettre de visualiser les informations contenues dans le cube, de sélectionner un sous-cube, de modifier l'ensemble des dimensions à prendre en compte ou de changer leur granularité. Comme mentionné précédemment, ces opérateurs de navigation sont généralement décomposés en trois catégories : opérateurs de structuration (Rotate, Switch, Push, Pull), de sélection (Slice, Dice) et d'agrégation (Roll-up, Drill-down). Dans cette partie, nous décrivons rapidement les opérateurs les plus utilisés dans la littérature et qui représentent un intérêt pour la suite de notre travail.

1.2.5.1 Opérateurs de sélection : *Slice* et *Dice*

L'opérateur *Slice* (littéralement, trancher), permet de sélectionner un sous ensemble du cube, selon une ou plusieurs valeurs d'une dimension particulière. *Slice* est un opérateur de sélection portant sur les valeurs d'une seule dimension. La Figure.1.5 montre un exemple de *Slice* : à partir du cube "Rendement Agricole" initial, on sélectionne un sous-ensemble de ce cube tel que la "Localisation" soit "France" ou "USA". L'opérateur *Dice* permet de faire une projection du cube. L'application de *Dice* équivaut à l'application de l'opérateur *Slice* à plusieurs dimensions. La Figure.1.6 montre un exemple de *Dice* : à partir du cube "Rendement Agricole" initial, on sélectionne un sous-ensemble de ce cube tel que la "Localisation" soit "France" ou "USA", et que la "Date" soit "2009" ou "2011". *Slice* ne peut produire que des tranches du cube, du fait qu'une sélection n'est possible que sur une dimension, alors que *Dice* peut produire des sous-cubes arbitraires.

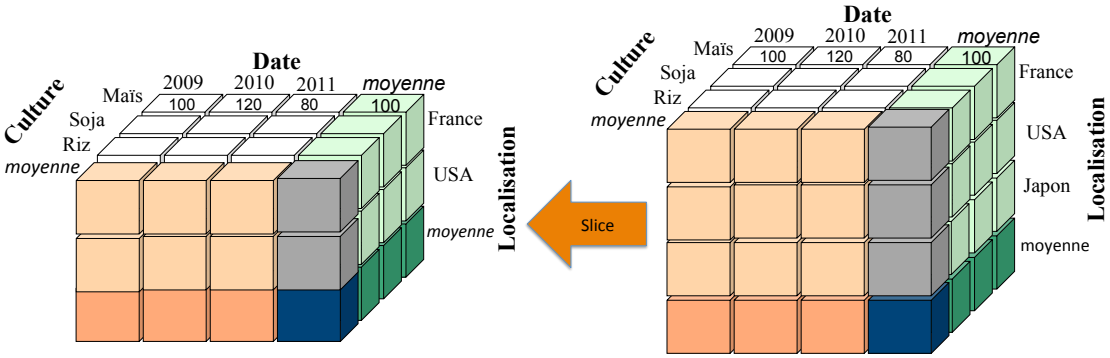


FIGURE 1.5 – Opérateur OLAP : *Slice*

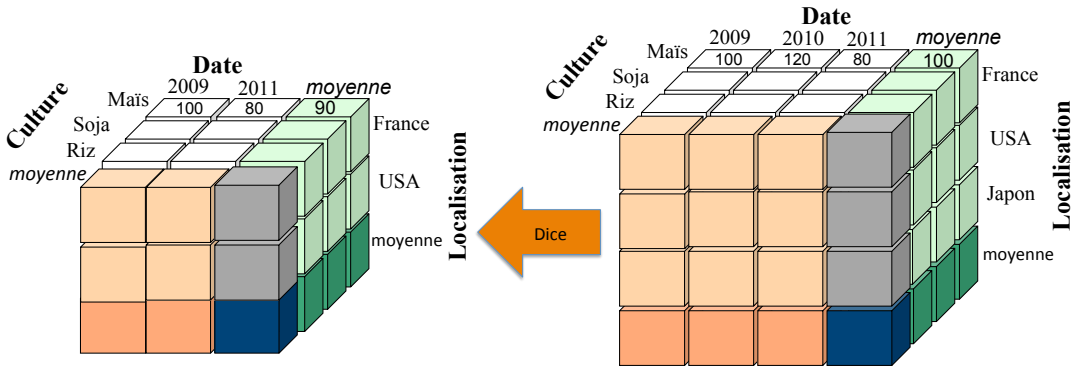


FIGURE 1.6 – Opérateur OLAP : *Dice*

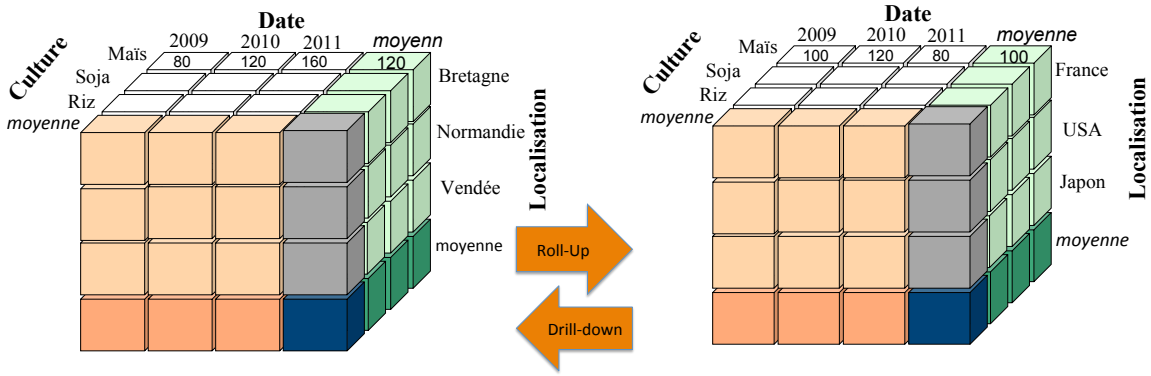


FIGURE 1.7 – Opérateurs OLAP : *Roll-up* et *Drill-down*

1.2.5.2 Opérateurs d'agrégation : *Roll-up* et *Drill-down*

Ce sont des opérateurs servant à naviguer entre différentes granularités d'une ou de plusieurs dimensions. Le *Drill-down* permet d'afficher les données avec une granularité plus fine. Un exemple de *Drill-down* est présenté dans la Figure.1.7 sur le cube "Rendement Agricole". Sur cet exemple, l'utilisateur effectue un *Drill-down* sur la valeur "France" de la dimension "Localisation". Le cube résultat présente seulement les régions ayant pour pays "France". Le *Roll-up* est l'inverse de *Drill-down* et consiste à remonter d'un niveau dans une hiérarchie de dimension vers un niveau plus agrégé. Dans le deux cas, le nombre de dimensions du cube reste inchangé, en d'autres termes, l'application d'un *Roll-up*/*Drill-down* sur un cube à n dimensions résulte en un cube à n dimensions. La Figure.1.7 présente l'effet d'un *Roll-up* sur la dimension "Localisation" du cube "Rendement Agricole".

Dans la sous-section suivante, nous citons quelques uns des domaines d'application des entrepôts de données en mettant l'accent sur le domaine de l'agro-environnement.

1.2.6 Domaines d'application des entrepôts de données

Les entrepôts de données, originellement utilisés dans le domaine du commerce et de la gestion, commencent à l'être dans divers autres domaines, comme par exemple les applications scientifiques (ex. la biologie, imagerie, génétique,...) . Cependant, peu d'entrepôts de données [Abd09, NSR08, PMB+10] ont été développées dans le domaine des sciences agro-environnementales. Pourtant, ils constituent un support pour l'analyse en ligne des données agricoles associées à de multiples dimensions.

Abdullah et al. [Abd09] ont développé un outil d'analyse en ligne, *ADSS-OLAP*, pour analyser l'incidence de la cochenille sur les cultures de coton. Une autre étude [PMB+10], préconise l'utilisation du langage de modélisation unifié (UML) pour modéliser les entrepôts de données agricoles. Cette méthode est ensuite appliquée pour analyser l'impact spatial de l'utilisation des pesticides dans l'agriculture.

Nilakanta et al. [NSR08] ont proposé l'entrepôt de données *INARIS (National Agricultural Resources Information System)* pour le secteur agricole indien. Cet entrepôt de données fournit des informations stratégiques et périodiques aux chercheurs et aux planificateurs pour améliorer le processus d'aide à la décision. Cependant, il s'avère difficile de développer une discussion critique sur les fonctionnalités de ces outils, vu que les entrepôts de données agro-environnementaux demeurent peu développés. Nous pensons que cela est dû principalement au fait qu'il est difficile de recueillir des données de terrain dans les sciences agro-environnementales. En effet, la collecte de données dans ces domaines de recherche reste un processus lent et coûteux.

1.3 Enrichissement de l'analyse OLAP

La grande capacité de stockage, la modélisation multidimensionnelle des données et les opérateurs OLAP font de l'entrepôt de données une plate-forme décisionnelle servant à l'analyse, la visualisation et la navigation dans les grandes masses de données. Étant donné que la taille des entrepôts de données augmente constamment et considérablement, le besoin de méthodes et d'outils qui permettent d'automatiser le processus d'extraction de connaissances, ou de guider l'utilisateur vers les sous-ensembles de données présentant un intérêt particulier, devient évident. La technologie OLAP se limitant à des tâches exploratoires, c'est donc à l'utilisateur de trouver manuellement les connaissances potentiellement contenues dans les données d'un cube (tâche très fastidieuse en présence de gros volumes de données).

À partir de ce constat, l'idée a émergé de faire évoluer l'OLAP vers d'autres possibilités d'analyse, et de fournir à l'utilisateur des outils automatiques pour l'aider à expliquer par exemple : des valeurs de cellules, des associations existant entre les données multidimensionnelles, ou à prédire des valeurs dans le cube, ...etc. La fouille de données peut apporter des réponses à ces besoins. En effet, la fouille de données représente un ensemble de techniques et méthodes destinées à extraire des informations intéressantes (non triviales, implicites, non connues précédemment et potentiellement utiles), des règles, des contraintes ou des motifs, à partir de gros volumes de données. Une fois validée, l'information extraite devient une connaissance. Dans ce qui suit, nous présentons quelques études menées dans le cadre du couplage entre la fouille de données et l'analyse en ligne.

1.3.1 Couplage fouille de données et analyse en ligne

Divers travaux de recherche [Han97, SS01, Sar99, CZC01, PHP+01, BF09a, BNBMLRB08] se sont intéressés à la combinaison de la fouille de données et de l'analyse en ligne. Le concept de fouille de données multidimensionnelles, ou *OLAP mining*, a été introduit par Han dans [Han97]. Il correspond à un processus de fouille de données (*data mining*) intégrant une composante OLAP, c'est-à-dire s'effectuant à partir de données multidimensionnelles.

Cependant le couplage de l'analyse en ligne et de la fouille de données n'est pas aisé. Plusieurs verrous scientifiques sont à lever : intégration des algorithmes de fouille

de données dans l'OLAP et les entrepôts de données ; stockage dans un entrepôt de données des connaissances extraites par une méthode de fouille de données ; exécution en ligne, sur des cubes parfois volumineux, d'algorithmes de fouille de données parfois coûteux en temps de calcul, etc.

Dans la littérature [MRBB04], différentes méthodes ont été proposées pour la mise en oeuvre du couplage de la fouille de données et de l'analyse en ligne. Trois grandes approches ont été mises en oeuvre :

- L'extension de l'analyse OLAP et des langages de requêtes : cette approche est issue principalement des travaux de [Han97, Han98] sur *DBMiner*. Elle consiste à étendre le langage des requêtes OLAP pour simuler des techniques de fouille de données telles que l'extraction de règles d'association, la classification, le clustering, la prédiction, ...etc. [Han97, SS01, Sar99, GC01, CDH00, BF09a],
- L'adaptation des structures multidimensionnelles : cette approche transforme les données multidimensionnelles afin de les rendre exploitables par les méthodes de fouille de données. C'est l'approche la plus intuitive mais elle peut être réductrice car l'aspect hiérarchique des données est perdu (i.e. toutes les dimensions sont aplaties), [GC01, CZC01, PHP$^+$01],
- L'adaptation des algorithmes de fouille de données : cette dernière approche consiste à modifier les algorithmes de fouille de données afin de pouvoir les utiliser directement dans un environnement multidimensionnel et hiérarchique, [SAM98, GMN08, BNBMLRB08].

Nous allons maintenant présenter une synthèse bibliographique des différents travaux associant l'aspect exploratoire de l'analyse en ligne à la démarche descriptive et prédictive de la fouille de données. Nous avons regroupé ces travaux selon le type (symbolique ou statistique) de méthodes de fouille de données utilisé.

1.3.1.1 Couplage fouille de données symboliques et analyse en ligne

Les premières tentatives d'extension des outils OLAP avec des méthodes de fouille de données remontent à 1997 avec les travaux de Han [Han97]. Ces travaux ont abouti à la création du système *DBMiner*. Ce système est fondé sur une approche M-OLAP. Les données sont stockées dans des structures multidimensionnelles, et plusieurs modules d'analyse sont proposés. Ces modules correspondent à des opérateurs OLAP auxquels on a ajouté des extensions permettant de simuler diverses techniques de fouille de données. Cependant, les références relatives à *DBMiner* décrivent plutôt le côté fonctionnel de ce dernier et ne donnent pas assez d'information sur les procédés employés et la méthodologie théorique suivie.

Par la suite, divers travaux se ont intéressés à l'exploration automatique des cubes de données avec des algorithmes d'extraction de règles d'association [MHC97, GC98, IKA02]. Les différents auteurs (Kamber, Imieliski, Goil, Choudhary, ...) exploitent la

structure multidimensionnelle du cube, avec ses agrégats pré-calculés, qu'ils considèrent comme un contexte favorable pour la recherche de règles. En effet, le support et la confiance des règles peuvent être calculés directement à partir des fréquences obtenues avec la fonction d'agrégation *COUNT* pour le calcul des agrégats. Ils évitent ainsi de parcourir plusieurs fois toutes les données. Ainsi, selon cette approche, des opérateurs OLAP sont utilisés comme outils pour extraire et calculer les différents critères d'une règle à différents niveaux de granularité des dimensions du cube de données.

Plus récemment, l'auteur de [Lou11] a proposé un nouvel opérateur *AROX (Association Rules Operator for eXplanation)* pour expliquer des phénomènes observés dans le cube à l'aide de règles d'association. En effet, l'opérateur *AROX* se base sur une recherche guidée de règles d'association. Afin d'adapter le support et la confiance d'une règle au contexte OLAP, l'auteur propose une nouvelle définition du support et de la confiance en y associant les mesures quantitatives du cube de données. Ainsi, contrairement à toutes les autres approches, une règle d'association n'est plus évaluée selon le nombre d'occurrences des faits qu'elle supporte mais selon les mesures des faits qu'elle supporte. Par exemple, un agriculteur est plus intéressé d'analyser ses cultures en fonction de leur rendement plutôt qu'en fonction de la surface qu'elles occupent.

D'autres études [PLL+10, NQN04, WK06, PHP+01] se sont intéressées à l'extraction de motifs fréquents dans le contexte de données multidimensionnelles et hiérarchiques. Les auteurs de [NQN04] Naouali et al. proposent d'extraire des motifs fréquents à partir de la table de fait d'un cube de données, où donc chaque motif représente un fait au sens OLAP. Selon les auteurs, les motifs fréquents permettent de mettre en évidence des liens sémantiques traduisant des relations intéressantes entre les cellules du cube étudié. Ces liens sont alors basés sur les ensembles fréquents que se partagent les cellules du cube. Les auteurs de [PLL+10, PHP+01] se sont intéressés quant à eux à la combinaison de plusieurs dimensions d'analyse et à la prise en compte des hiérarchies lors du processus d'extraction de motifs séquentiels multidimensionnels. Ceci afin de permettre une extraction de connaissances plus complète et dont l'utilisation dans le contexte OLAP peut être envisageable.

1.3.1.2 Couplage fouille de données statistiques et analyse en ligne

D'autres tentatives d'extension des capacités de l'OLAP ont émergé [SAM98, YGJ+06, CRST06, Lou11] avec la volonté d'étendre l'OLAP à la prédiction en ligne (i.e. analyse de type *What If Analysis*). Dans ce contexte, le couplage entre l'OLAP et la fouille de données permet de prédire la valeur de la mesure pour des faits inexistants ou des faits avec une valeur manquante. L'objectif est de permettre à l'utilisateur d'analyser les données du passé mais aussi d'anticiper les événements du futur. La plupart de ces travaux, utilisent comme méthode de prédiction un modèle de régression (linéaire, log-linéaire, logistique, ...).

D'autres études [MRBB04, BF09b] ont tenté de relâcher la contrainte du schéma fixe

de l'entrepôt en permettant la modification de la structure hiérarchique d'une dimension. Ces travaux se basent sur des méthodes de classifications pour définir de nouvelles hiérarchies basées sur des relations sémantiques. Les auteurs de [MRBB04] ont proposé un nouvel opérateur OLAP, baptisé *OpAC (Operator for Aggregation by Clustering)*, basé sur une méthode *CAH (Classification Ascendante Hiérarchique)*. L'opérateur *OpAC* consiste en l'agrégation sémantique des valeurs d'une dimension d'un cube de données en se basant sur la technique *CAH*. Les opérateurs OLAP classiques agrègent les valeurs d'une dimension selon des liens hiérarchiques prédéfinis. En revanche, *OpAC* permet de créer de nouveaux agrégats qui reflètent des fait réels en exploitant les mesures contenues dans le cube de données.

Les auteurs de [BF09b] ont quant à eux proposé un nouvel opérateur d'agrégation *RoK (Roll-up with K-means)* en utilisant la méthode de classification automatique des *k-means*, qui permet de rechercher des structures naturelles dans les données. L'opérateur *RoK* permet de créer un nouveau niveau de granularité dans une hiérarchie de dimension en se basant sur les *K-means*. Il s'agit de trouver un bon regroupement des instances d'un niveau d'analyse existant choisi par l'utilisateur, à partir duquel un nouveau niveau d'analyse peut être créé. Cette approche enrichit l'analyse multidimensionnelle en offrant de nouveaux angles de vues intéressants sur les faits pouvant être explorés par l'utilisateur.

Afin d'assister l'utilisateur dans sa tâche d'exploration des cubes de données, certains travaux [SAM98, Lou11, LB12, Sar01] ont proposé des méthodes basées sur des modèles statistiques et notamment d'analyse factorielle pour le guider vers les régions intéressantes du cube et pour réorganiser intelligemment les dimensions du cube. Dans [SAM98], Sarawagi et al. proposent un outil d'identification des régions remarquables dans les cubes de données. Ils proposent une exploration guidée par la découverte (*Discovery-Driven*). Leur idée consiste à intégrer un module statistique de régression multidimensionnelle dans un serveur OLAP en vue de guider l'utilisateur pour détecter des valeurs outliers à différents niveaux hiérarchiques d'un cube de données. Une amélioration de ces travaux a été proposée dans [Sar01]. Cette amélioration concerne une meilleure automatisation de l'analyse par l'emploi de la programmation dynamique. Cette automatisation est garantie par un nouvel opérateur, appelé *iDiff*, qui détecte les régions outliers et explore les raisons de la présence de ces régions dans un cube de données.

Plus récemment de nouveaux opérateurs (*VOCoDa et ORCA*) [Lou11, LB12, BMBLR07, MBR06] ont été proposés pour l'analyse en ligne de données complexes (données multi-sources, multi-formats et multi-structures). Ces deux opérateurs se basent sur des méthodes factorielles (AFC et ACM) pour la visualisation des faits dans un cube et la détection de régions intéressantes en réorganisant les dimensions du cube.

L'opérateur *VOCoDa (Visualization Operator for Complex Data)* a été proposé par les auteurs de [LB12]. Ils utilisent les principes de l'Analyse Factorielle des Correspondances (AFC), pour visualiser des données complexes tout en tenant compte de leurs proximités ou de leurs différences, en intégrant donc une partie de la sémantique contenue dans les données. L'analyse des correspondances produit des axes factoriels qui

peuvent être utilisés comme de nouvelles dimensions. Ces nouvelles dimensions constituent un nouvel espace de représentation dans lequel il est possible de projeter les faits et de mettre en évidence des points de vue intéressants pour l'analyse.

L'opérateur *ORCA (Operator for Reorganization by multiple Correspondence Analysis)* [BMBLR07, MBR06, Lou11] quant à lui permet de réorganiser des données multidimensionnelles pour détecter des régions intéressantes dans le cube en exploitant les principes de l'Analyse des Correspondances Multiples (ACM). Les auteurs préconisent d'utiliser l'ACM car cette méthode descriptive de fouille de données fournit une représentation graphique synthétique d'une grande quantité de données décrites par des variables qualitatives. En effet, cet opérateur réorganise les faits dans l'espace de représentation que constitue le cube et est d'autant plus efficace lorsque ce dernier est volumineux et épars. Les axes factoriels obtenus par l'ACM donne une autre disposition du cube dans laquelle les faits sont beaucoup moins éparpillés et regroupés selon leur contenu.

Ces deux opérateurs d'analyse en ligne des données complexes sont implémentés dans la plate-forme *MiningCubes* [Lou11].

1.3.2 Couplage recherche d'information et analyse en ligne

On constate que de nombreuses études ont exploré la piste du couplage de l'OLAP et de la fouille de données pour l'enrichissement de l'analyse en ligne. Cependant, l'idée d'associer l'OLAP à la fouille de données ne suffit pas car elle ne couvre évidemment pas tous les problèmes liés aux limites de l'analyse en ligne. En effet, l'avènement des données complexes laisse entrevoir de nouveaux verrous scientifiques à relever. Par exemple, les données complexes peuvent être composées de documents qui contiennent, entre autres, du texte. Dans ce cas, comment modéliser de façon multidimensionnelle de telles données et comment faire une analyse en ligne de ces dernières. La communauté scientifique, en particulier celle des bases de données, a cherché une voix autre que le couplage de l'OLAP et de la fouille de données pour répondre à ces nouveaux besoins. Certaines études [JGO02, MLC+00, PP03, MCDA03] se sont alors intéressées à l'association de la recherche d'information (en particulier des techniques de traitement de documents textes) à l'analyse en ligne. La recherche d'information cherche des documents qui répondent à un besoin d'information d'un utilisateur exprimé à l'aide d'une requête. L'utilisateur décrit à l'aide de mots-clés l'information qu'il cherche. Le système de recherche d'information évalue les documents par rapport à l'information recherchée, et retourne les documents jugés comme les plus pertinents. Depuis une dizaine d'années, quelques équipes de recherche s'intéressent au couplage de l'OLAP et de la recherche d'information. Les premiers travaux remontent au début des années 2000 [JGO02, MLC+00]. Plus récemment, l'équipe de Han a introduit le concept de base de données textuelles multidimensionnelles (*multidimensional text database*) en le distinguant des bases de données relationnelles et des bases de documents textes [YSC+09, BBCX+10].

Compte tenu de notre problématique portant sur des entrepôts de données non complexes (i.e. données numériques et/ou catégoriques), nous n'allons pas détailler ces méthodes.

1.3.3 Discussion et conclusion

Les différents travaux présentés ci-dessous sont une démonstration de l'intérêt du couplage de l'OLAP à d'autres techniques comme la fouille de données et la recherche d'information pour faire significativement évoluer l'analyse en ligne. Elles sont une partie des réponses possibles à cette problématique. Mais il existe beaucoup de méthodes de fouille de données, de recherche d'information ainsi que d'autres méthodes d'analyse. L'idée n'est pas forcément de combiner toutes ces possibilités à l'OLAP mais de réfléchir à l'évolution que nous souhaitons de l'analyse en ligne. En effet, les techniques de fouille de données et de recherche d'information visent des objectifs d'analyse différents : visualisation, description, structuration, classification, explication, prédiction,...etc. Par exemple, les techniques d'extraction de règles d'association et de motifs séquentiels se focalisent sur des possibilités d'analyse de type : explication et prédiction. Les méthodes de classification quant à elles visent une analyse basée sur la classification et la structuration des données multidimensionnelles. Les modèles de régression (linéaire, log-linéaire, logistique, ...) sont des méthodes de prédiction. Pour finir, l'analyse factorielle est elle dédiée à la visualisation et à la description des données.

Nous avons aussi vu qu'il existe trois approches différentes de couplage de l'analyse en ligne avec la fouille de données. Ces approches sont aussi valables pour le couplage de l'analyse en ligne avec la recherche d'information, vu que les deux couplages présentent les mêmes verrous techniques. La première approche préconise l'exploitation ou l'extension des outils existants à des tâches de fouille de données. Cette possibilité est intéressante mais ne couvre pas toutes les méthodes. La deuxième approche consiste à transformer des données multidimensionnelles en données tabulaires. Cette approche consiste en l'adaptation des algorithmes de fouille aux données multidimensionnelles. Elle est simple et intuitive mais présente le risque de faire perdre aux données transformées leur aspect hiérarchique. À notre avis, cette voie est la plus prometteuse mais la plus difficile à réaliser.

Dans notre étude, nous manipulons des données issues de résultats de simulations. Ces données sont complètes (pas de données manquantes), ce qui engendre un cube de données volumineux et dense. Par ailleurs, l'analyse en ligne classique ne dispose pas d'outils permettant de guider l'utilisateur vers les faits les plus intéressants du cube. C'est à l'utilisateur de manipuler au mieux le cube de données pour y découvrir des zones d'informations pertinentes. Dans un cube de données volumineux, cette navigation peut s'avérer très fastidieuse. Par conséquent, le recours à des méthodes spécifiques pour détecter automatiquement les zones susceptibles d'intéresser l'utilisateur augmenteraient de façon significative le pouvoir analytique de l'OLAP. Guider l'utilisateur dans son exploration des données et l'aider à pouvoir intégrer ses connaissances dans l'entrepôt suppose une prise en compte par le système de ses préférences. Parmi les techniques citées dans la section précédente, seules les méthodes basées sur les modèles statistiques (analyse factorielle (*VOCoDa et ORCA*) et régression multidimensionnelles (*iDiff*)) permettent de guider l'utilisateur dans sa navigation, soit en identifiant des régions remarquables dans les cubes de données, soit en réorganisant les données multidimensionnelles pour détecter des régions intéressantes. Cependant, aucune de ces méthodes

n'intègre les préférences des utilisateurs afin de leur fournir des résultats focalisés sur leurs centres d'intérêt.

Quelques travaux se sont intéressés à l'intégration des préférences utilisateurs dans le processus d'analyse en ligne. Bellatreche et al. [BGM+05] se sont inspirés des techniques de filtrage d'information en fonction du profil utilisateur pour affiner des requêtes en y ajoutant des prédicats. L'objectif de ces travaux est de fournir à l'utilisateur un résultat conforme à ses préférences et à ses attentes, tout en prenant en compte des contraintes de visualisation personnalisées. Cette méthode implique donc la gestion de profils utilisateurs. Ravat et al. [RT09] proposent une approche de personnalisation des données multidimensionnelles manipulées, pour aider l'utilisateur dans l'exploration de ses données. Un poids fixé par l'utilisateur est associé aux données multidimensionnelles lui permettant d'exprimer ses préférences. Ceci permet de générer uniquement les données identifiées comme pertinentes en fonction des poids. L'inconvénient de cette approche, réside dans le fait que ces préférences (poids) sont exprimées de manière absolue. Pour pallier ce problème, Jerbi et al. [JRTZ08] proposent une autre solution basée cette fois sur l'exploitation d'ordre de préférences et non plus de poids. Ces ordres ne sont pas exprimés de façon absolue, mais par rapport à un contexte d'analyse donné. Par conséquent, les préférences peuvent varier d'un contexte d'analyse à l'autre. Ce qui engendre un système d'analyse beaucoup plus flexible (i.e. permet une gestion dynamique des préférences utilisateurs). Pour terminer, nous pouvons citer, d'un point de vue un peu plus formel, la proposition de Golfarelli et Rizzi [GR09] d'une algèbre permettant la prise en compte de préférences pour l'analyse en ligne. Ces préférences peuvent porter sur les mesures, les attributs des dimensions ou les hiérarchies elles-mêmes. Cependant, la plupart de ces travaux ne prennent pas en compte les préférences associées à des données catégoriques. Aussi, en présence de préférences conflictuelles, les requête OLAP ne retournent aucun résultat à l'utilisateur, vu qu'aucune donnée ne correspond à cet ensemble de préférences. Les requêtes OLAP ne sont pas adaptées à ce type d'analyse. Pourtant, dans un contexte de systèmes d'aide à la décision comme les entrepôts de données, il serait très pertinent de permettre à l'utilisateur d'évaluer plusieurs options dans des situations où aucune solution ne s'impose à lui.

Dans les systèmes interactifs d'aide à la décision, l'aide à la décision est associée à l'analyse multi-critères (méthode de recherche d'information). Une telle analyse vise à répondre à différents objectifs le plus souvent contradictoires afin d'aider à prendre une décision. Parmi les différentes méthodes d'analyse multi-critères qui existent, nous portons une attention particulière aux *Requêtes Skyline* [BKS01]. En effet, les requêtes skyline sont des requêtes avec préférences. Elles permettent de sélectionner à partir d'une base de données volumineuse, les meilleurs tuples dans le sens de la préférence. Les requêtes répondent complètement aux besoins que nous avons exprimés. Nous proposons donc de coupler les requêtes skyline avec l'analyse en ligne. Les requêtes skyline étant des requêtes multi-critères (i.e. multidimensionnelles), elles s'adaptent très bien à l'aspect multidimensionnel des entrepôts de données et donc de l'analyse OLAP. Comme approche de couplage, nous choisissons l'adaptation des algorithmes d'extraction des requêtes skyline à l'aspect hiérarchique des données. Nous proposons deux méthodes d'extraction de skyline dans le contexte des entrepôts de données. La première mé-

thode, EC^2SKY [BCQ12], permet un calcul incrémental des skyline en présence de préférences dynamiques. Les préférences dynamiques sont des préférences qui changent d'un utilisateur à un autre. La seconde, $HSky$, permet à l'utilisateur de naviguer tout au long des dimensions hiérarchiques (i.e. drill-down/ roll-up), tout en assurant un calcul en ligne des skyline. Lors de la navigation le long des axes des dimensions, l'utilisateur a la possibilité de spécialiser ou de généraliser ces préférences. Ces deux contributions seront détaillées dans les prochains chapitres (4 et 5). Dans le prochain chapitre, après avoir défini formellement le problème de la recherche des skyline, nous donnerons un aperçu des différentes méthodes d'extraction de points skyline dans le contexte des bases de donne ?es.

Chapitre 2

Analyse multicritères dans les bases de données

2.1 Introduction

L'extraction et la mise en évidence de données pertinentes est une tâche difficile en particulier lorsque les utilisateurs sont confrontés à de gros volumes de données pouvant être comparées selon de nombreux critères. Dans le contexte des bases de données décisionnelles, lorsque ces critères (i.e. dimensions) sont conflictuels, les requêtes d'interrogation peuvent être infructueuses. En effet, un point (i.e un tuple) peut être optimal pour un critère mais pas pour un autre. Il est alors éliminé du résultat alors qu'il aurait pu être pertinent pour l'utilisateur. Afin d'apporter une réponse adéquate à ce type de requêtes, le concept de *requêtes skyline* a été introduit par les auteurs de [BKS01].

Divers travaux [RPK10, WFP+08, WPFW09, YLL+05, TXP08, JTEH07, HGSW08, BKS01, SZJ07] se sont intéressés à l'extraction des points skyline comme outil de restitution dans un contexte décisionnel. Les requêtes skyline permettent de formuler des requêtes multi-critères [SNT85] et d'extraire les points globalement optimaux pour l'ensemble des critères considérés. Par exemple, imaginez un vacancier à la recherche d'un hôtel. Il se rend dans une agence de voyage et demande un hôtel qui soit à la fois près de la mer et peu onéreux. Malheureusement, ces deux objectifs sont contradictoires car généralement, plus les hôtels sont près de la plage plus ils sont chers. La base de données utilisée par l'agence de voyage ne permet pas d'identifier facilement quel hôtel sera le meilleur pour le vacancier mais elle permet d'identifier des hôtels "intéressants". Il s'agit d'hôtels qui sont meilleurs que les autres hôtels sur les deux dimensions. Ces hôtels les plus intéressants font partie d'un ensemble appelé skyline. A partir de cet ensemble d'hôtels, le vacancier peut faire son choix.

La recherche des skyline est un problème d'optimisation multi-critères classique [Ste86, SNT85], aussi connu sous le nom d'optimisation multi-objectifs ou multi-attributs. Ce problème a de nombreuses applications dans le monde réel. En effet, les décideurs de nombreux domaines d'activités (industrie, agriculture, finance, ...) doivent rechercher

les meilleurs compromis afin d'atteindre de multiples objectifs parfois conflictuels. Dans de nombreux cas, ces problèmes de décision du monde réel peuvent être formulés sous la forme mathématique des problèmes d'optimisation multicritères. Pour résoudre ce problème, si une solution unique minimisant simultanément tous les objectifs n'existe pas (il n'y a pas d'hôtel qui soit à la fois le moins cher et le plus près de la plage), on cherche à effectuer le meilleur choix possible entre plusieurs solutions en optimisant simultanément 2 à n critères conflictuels, sujets à certaines contraintes. Il s'agit donc de rechercher une (ou plusieurs) solution(s) pour laquelle (lesquelles) chaque objectif a été optimisé au maximum dans la mesure où si l'on essaye d'optimiser encore ce critère, un autre critère va en pâtir. On obtient ainsi un ensemble de solutions et l'on est capable de quantifier en quoi ces solutions sont meilleures que les autres. La recherche de ces solutions requiert des techniques appropriées et efficaces.

Les bases de la théorie de la décision multicritères ont été posées par Pareto à la fin du 19ème siècle et depuis on peut recenser de nombreux travaux [HFZT08, Deb08, Mie05, GS06] sur des problèmes proches comme la théorie d'optimisation des vecteurs (Theory of Vector Optimization), la programmation non linéaire multi-objectifs (Nonlinear Multiobjective Programming), la programmation multi-objectifs floue (Fuzzy Multiobjective Programming), l'optimisation combinatoire multi-objectifs (Multiobjective Combinatorial Optimization), les problèmes d'emplois du temps multi-critères (Multicriteria Scheduling Problems). On trouve également de nombreux travaux qui se consacrent aux différentes méthodologies pouvant être utilisées pour résoudre ces problèmes : Goal Programming, Interactive Methods, Evolutionary Algorithms, Data Envelopment Analysis.

En ce qui concerne la recherche des skyline, dans les années 60, ce problème était connu comme le problème de la recherche des points admissibles [BM79] ou des vecteurs maximaux [BKST78]. Ces premiers algorithmes se sont montrés inefficaces dans le contexte des grandes bases de données, contenant de nombreux points et de nombreux critères. Les auteurs de [BKS01] furent les premiers à proposer des solutions dans ce contexte en intégrant un opérateur skyline étendant le langage SQL. Désormais de nombreux outils d'aide à la décision intègrent des méthodes de recherche de skyline. Dans ce chapitre, après avoir défini formellement le problème de la recherche des skyline, nous donnerons un aperçu des différentes méthodes d'extraction de points skyline dans le contexte des bases de données.

2.2 Requêtes skyline : concepts de base

Les requêtes skyline, constituent un paradigme élégant et sophistiqué pour des requêtes à préférences dans les bases de données multidimensionnelles. Elles ont été largement étudiées dans les communautés des bases de données et de l'intelligence artificielle. Dans un espace multidimensionnel où une préférence est définie pour chaque dimension, les requêtes skyline retournent les points qui sont meilleurs ou égaux sur toutes les dimensions et strictement meilleurs sur au moins une dimension.

Dans cette section, nous donnons quelques définitions préliminaires de concepts relatifs à la théorie de la préférence et à la relation de dominance. Dans un premier temps, nous définissons les propriétés d'une relation de préférence, et nous présentons les concepts relatifs aux requêtes skyline. Ensuite nous nous intéressons aux algorithmes de calcul des skyline dans le contexte des bases de données multidimensionnelles. Nous présentons ensuite les différents travaux de recherche liés à l'extraction des skyline dans des sous-ensemble de dimensions. Dans un second temps, nous discutons des travaux existants sur le calcul des points skyline en présence de dimensions hiérarchiques et de données agrégées.

2.2.1 Ordres de préférence et skyline

Les notations utilisées dans cette section sont récapitulées dans la table 2.2. Les différentes définitions sont illustrées à partir de l'exemple de la table 2.1 qui décrit des propositions de voyage selon les dimensions suivantes : le prix, la distance de la plage et le groupe de l'hôtel. Nous allons illustrer les définitions de dominance, de skyline et de relation de préférence sur cet exemple :

Package ID	Prix(Pr)	Distance(Dis)	Groupe(Gr)
a	1600	4	T (Tulips)
b	2400	1	T(Tulips))
c	3000	5	H(Horizon)
d	3600	4	H(Horizon)
e	2300	2	T(Tulips)
f	3000	3	M(Mozilla))
g	3600	4	M(Mozilla)

TABLE 2.1 – Base ensemble d'hôtels

Soit $D = \{d_1, ..., d_n\}$ un espace à n dimensions, E un ensemble de points définis dans l'espace D et p, q deux points de l'ensemble E. On note par $p(d_i)$ la valeur de p sur la dimension d_i. Une préférence sur le domaine de valeurs de la dimension d_i, notée par $\wp_{d_i}$, est définie par un ordre partiel $\preccurlyeq_{d_i}$. Elle désigne l'ensemble des préférences binaires $\wp_{d_i} = \{(u, v) | u \preccurlyeq_{d_i} v\}$, avec (u, v) un couple de valeurs. Dans la suite, nous utilisons les deux notations pour $\wp_{d_i}$.

Notation	Description
E	Ensemble de données
$\|E\|$	Cardinal de l'ensemble E
D	Ensemble des dimensions de E
$\|D\|$	Cardinal de D
d_i	une dimensions de l'espace D ($1 \leqslant i \leqslant \|D\|$)
D'	sous espace de D : $D' \subseteq D$
p, q	Points de l'ensemble E
$p(d_i)$	Valeur du point p sur la dimension d_i
$dom(d_i)$	Domaine de valeurs de d_i
$\wp$	Ensemble de préférences sur D'
$\wp_{d_i}$	Préférence sur d_i

TABLE 2.2 – Récapitulatif des notations

Définition 1 ***(Relation de préférence)*** *Une relation de préférence sur le domaine de valeurs d'une dimension d_i est définie par un ordre partiel noté $\leqslant_{d_i}$. Pour deux valeurs $p(d_i)$ et $q(d_i)$ du domaine de valeurs de d_i, nous écrivons $p(d_i) \leqslant_{d_i} q(d_i)$ si la valeur $p(d_i)$ est préférée à la valeur $q(d_i)$.*

Définition 2 *Étant donné une relation de préférence $\leqslant_{d_i}$, les relations correspondantes de préférence stricte, indifférence et incomparabilité sont définies comme suit :*

- $p <_{d_i} q$ *(p est strictement préféré à q sur la dimension d_i)* $\Leftrightarrow p \leqslant_{d_i} q \wedge \neg(q \leqslant_{d_i} p)$
- $p =_{d_i} q$ *(p est indifférent (i.e. également préféré) à q sur la dimension d_i)* $\Leftrightarrow p \leqslant_{d_i} q \wedge q \leqslant_{d_i} p$
- $p \leqslant_{d_i} q$ *(p est préféré à q sur la dimension d_i)* $\Leftrightarrow p =_{d_i} q \vee p <_{d_i} q$
- $p \sim_{d_i} q$ *(p est incomparable à q sur la dimension d_i)* $\Leftrightarrow \neg(p \leqslant_{d_i} q) \wedge \neg(q \leqslant_{d_i} p)$

Exemple 4 *L'ensemble de données E décrit dans la table 2.1 contient 7 points décrits par 3 dimensions $Prix(Pr)$, $Distance(Dis)$ et $Groupe(Gr)$. Les valeurs des deux dimensions $Prix$ et $Distance$, sont totalement ordonnées par la relation d'ordre $\leqslant$ indiquant le plus petit de deux nombres réels. Cette préférence spécifie qu'on préfère les hôtels ayant le prix (resp. la distance) le moins élevé (ex. $a(Pr) <_{Pr} d(Pr)$). Les valeurs de la dimension $Groupe$ sont ordonnées selon la préférence suivante : $T <_{Gr} H$. Les autres valeurs de la dimension Groupe sont laissées non ordonnée (i.e. la valeur M pour cet exemple).*

Pour chaque dimension d_i dans l'espace D, une préférence est définie. Dans ce qui suit, $\wp = \bigcup_{i=1}^{|D|} \wp_{d_i}$ désigne l'ensemble des préférences associées à l'espace D. L'absence de préférence sur une dimension d_i (i.e. incomparabilité entre les valeurs de d_i) est notée par $\wp_{d_i} = \varnothing$. Dans le cas d'une indifférence ou d'une égalité (i.e. points ayant des valeurs identiques sur une dimension) entre deux points, ils peuvent être considéré comme équivalents et donc regroupés dans des classes d'équivalence. Il est aisé de constater

qu'avec l'utilisation des classes d'équivalence, la relation de préférence devient un ordre partiel strict sur une dimension.

Nous présentons maintenant les concepts et les définitions nécessaires à la définition formelle des skyline.

Définition 3 ***(Relation de dominance)*** *p domine q sur D, noté par $p \prec_D q$, si p est strictement préféré ou égal à q sur toutes les dimensions de D et p est préféré à q sur au moins une dimension : $\forall d_i \in D,\ p(d_i) \leqslant_{d_i} q(d_i) \wedge \exists d_i \in D,\ p(d_i) <_{d_i} q(d_i)$.*

Pour plus de lisibilité, $p \prec_D q$ sera simplement noté $p \prec q$.

Exemple 5 *Soit un client à la recherche d'un hôtel qui est à la fois proche de la plage et abordable. Dans ce cas, l'hôtel a domine l'hôtel d ($a \prec d$) since : $a(Pr) <_{Pr} d(Pr)$, $a(Gr) <_{Gr} d(Gr)$ et $a(Dis) =_{Dis} d(Dis)$.*
De plus, l'hôtel a ne domine pas l'hôtel b ($a \not\prec b$), car $b(Dis) <_{Dis} a(Dis)$ et $a(Pr) <_{Pr} b(Pr)$

L'ensemble des skyline, ou simplement le skyline, d'un ensemble de données décrit dans un espace de dimensions contient des points de cet ensemble de données qui ne sont dominées par aucun autre point de ce même ensemble.

Définition 4 ***(Skyline)*** *Le skyline de l'ensemble de données E sur l'espace de dimensions D, avec $\wp$ l'ensemble des préférences associées à D, est l'ensemble des points qui ne sont dominés par aucun autre point dans E :*
$Sky(D, E)_\wp = \{p \in E |\ (\forall q \in E,\ \neg(q \prec_D p))\}$.

Si $\wp = \varnothing$, $Sky(D, E)_\wp = E$.

Exemple 6 $Sky(D, E)_\wp = \{a, b, e\}$.

L'ensemble $Sky(D, E)_\wp$ contient des points notés $MaxSky(D, E)_\wp$ qui dominent tous les autres points sur au moins une dimension. Il contient aussi, et c'est là tout l'intérêt d'une telle approche, des points notés $CompSky(D, E)_\wp$ qui ne dominent sur aucune dimension de D tout en dominant les points $MaxSky(D, E)_\wp$ sur au moins une dimension. Ces points constituent une solution *compromis* intéressante pour l'utilisateur.

Définition 5 $Sky(D, E)_\wp = MaxSky(D, E)_\wp \bigcup CompSky(D, E)_\wp$ *avec :*
- $MaxSky(D, E)_\wp = \{p \in Sky(D, E)_\wp \ /\ \exists d_i \in D,\ \forall q \in E,\ q \leqslant_{d_i} p\}$
- $CompSky(D, E)_\wp = \{p \in Sky(D, E)_\wp \ /\ \forall q \in E,\ \exists d_i \in D,\ q <_{d_i} p\}$

Quand D est restreint à une seule dimension ($D = \{d\}$),
$Sky(D, E)_{\wp d} = MaxSky(D, E)_{\wp d}$.

Exemple 7 $Sky(, E)_\wp = \{a, b, e\}$.
$MaxSky(D, E)_\wp = \{a, b\}$ *car la valeur de a (resp. b) est préférée sur la dimension $Prix$ (resp. $Distance$).*
$CompSky(D, E)_\wp = \{e\}$ *car e n'est le meilleur sur aucune dimension, mais e est meilleur que b sur $Prix$ et e est meilleur que a sur $Distance$ (i.e., e constitue un bon compromis pour l'utilisateur) (c.f. Figure. 2.1).*

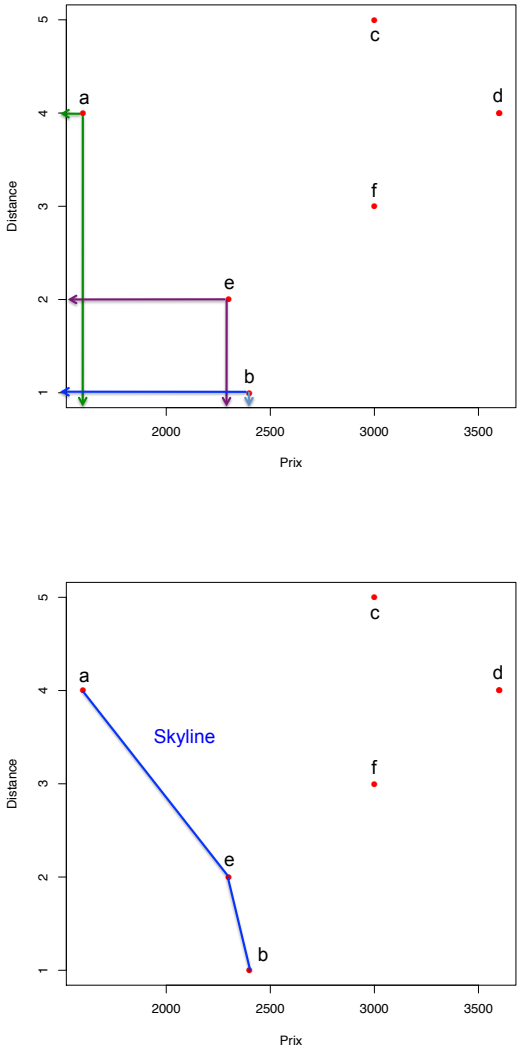


FIGURE 2.1 – Illustration du concept de dominance et de skyline sur les deux dimensions *Prix* et *Distance*

2.2.2 Propriétés des requêtes skyline

À présent, nous disposons d'une définition formelle du concept de requêtes skyline. Il faut maintenant évaluer l'efficacité de ces requêtes en examinant leur complexité. Pour calculer l'ensemble des points skyline, chaque point est comparé avec tous les autres points de l'ensemble de données (i.e. tests de dominance), ce qui représente $O(n^2)$ comparaisons (avec n le nombre de points dans l'ensemble de données). Dans le contexte de bases de données volumineuses, les calculs deviennent facilement intensifs (i.e. CPU-intensifs) en raison du grand nombre de tests de dominance. Il est donc nécessaire

d'utiliser des algorithmes spécifiquement conçus pour limiter le nombre de tests de dominance, et ainsi optimiser le calcul des skyline pour permettre le passage à l'échelle en présence de gros volumes de données.

Les requêtes skyline sont utilisées dans diverses applications, souvent dans des applications interactives. Il est important, dans ce type d'application, que l'algorithme de recherche de skyline utilisé fournissent des résultats rapidement voire instantanément. C'est un défi difficile et les algorithmes de calcul de skyline doivent satisfaire un ensemble de critères permettant l'évaluation de ces derniers en mettant en évidence leurs forces et leurs faiblesses. Ces critères d'évaluation ont été proposés par les auteurs de [DFS02, HAC+99]. Nous avons synthétisé ces critères en ne conservant que ceux qui nous semblent les plus pertinents dans le contexte de notre travail :

- **Efficacité** : Ce critère a trait à l'efficacité des algorithmes (i.e. temps de calculs, stockage mémoire ...etc.), mais aussi au passage à l'échelle de ces derniers (i.e. l'algorithme doit bien évoluer avec différentes distributions de données, différents types de données par exemple, numérique, catégoriques, et avec un nombre variable de dimensions, ...etc).
- **Progressivité** : Les points skyline sont retournés le plus rapidement possible, au fur à mesure qu'il sont découverts,
- **Préférences** : Le calcul des skyline intègre les préférences utilisateurs,
- **Correction** : Tous les points retournés sont des skyline,
- **Complétude** : Tous les points du skyline sont retournés à la fin.

Tous les critères décrits ci-dessus constituent les défis majeurs que doit relever un bon algorithme de calcul de skyline. Ces défis tant théoriques qu'expérimentaux, ont reçu une attention particulière de la part d'un grand nombre de chercheurs dans la communauté des bases de données et de l'intelligence artificielle. Dans la suite, nous présentons quelques uns des algorithmes qui ont été proposés pour le calcul et l'extraction des skyline. Une fois présentés, ces algorithmes seront évalués et comparés selon les critères décrits précédemment.

2.2.3 Algorithmes de calcul des skyline

Les requêtes skyline permettent l'extraction des points qui ne sont dominés par aucun autre point dans un espace multidimensionnel associé à des préférences. Divers travaux de recherche se sont intéressés au calcul des skyline en présence de larges volumes de données. Ces travaux peuvent être subdivisés en deux catégories : les algorithmes de calcul de skyline dans un espace complet ou dans des sous-espaces. Dans la première catégorie, les algorithmes s'intéressent à l'optimisation du calcul des skyline associés à toutes les dimensions de l'espace considéré, et ceux de la deuxième permettent à l'utilisateur de choisir les dimensions qui l'intéressent, tout en assurant un calcul optimal des skyline. Dans cette section, nous présentons les principaux algorithmes de chacune de ces catégories.

2.2.3.1 Algorithmes de recherche dans un espace complet

Nous distinguons deux familles d'algorithmes selon qu'ils utilisent ou non des techniques d'indexation. Les premiers n'utilisent pas de structure de données particulière (i.e. pas de pré-traitements), et donc la recherche des skyline se fait en analysant entièrement la base de données au moins une fois. Les seconds regroupent ceux qui exploitent des index afin d'améliorer l'efficacité de calcul et d'accélérer davantage les requêtes skyline, en diminuant par exemple le nombre de tests de dominance.

Les algorithmes sans index

L'algorithme Block Nested Loop (BNL) Börzsönyi et al. [BKS01] sont les premiers à avoir abordé la problématique du calcul des skyline dans le contexte des bases de données. Ils ont introduit l'opérateur SKYLINE dans les systèmes de gestion des basses de données, comme étant une clause SQL étendue. Ils ont aussi proposé dans [BKS01] le premier algorithme de recherche de skyline *Block Nested Loop* (BNL). Pour le calcul des skyline, une approche naïve serait de comparer tous les points de l'ensemble de données deux à deux et de retourner comme skyline tous les points qui ne sont dominés par aucun autre. L'algorithme BNL s'appuie sur cette approche en analysant entièrement l'ensemble des données et en gardant une liste de points candidats au skyline en mémoire centrale. Si la mémoire centrale est saturée, l'algorithme gère un fichier temporaire stocké en mémoire secondaire dans lequel sont stockés tous les candidats non considérés faute de place. Ils seront traités lors d'une prochaine itération. Lors de la comparaison d'un point p avec les points stockés en mémoire centrale, trois cas de figure se présentent :

- Le point p est dominé par un point présent en mémoire centrale : p est alors directement écarté de l'ensemble skyline (i.e. il est dominé) et ne sera plus pris en compte pour le reste des calculs,
- Le point p domine un ou plusieurs points présents en mémoire centrale : les points dominés sont directement écartés et ne seront plus pris en compte pour le reste des calculs (i.e. ils sont supprimés de la liste de points candidats au skyline). Le point p quant à lui est inséré en mémoire centrale, puisqu'il y a au moins une place libre,
- Le point p est incomparable avec l'ensemble des points présents en mémoire centrale : c'est le cas le plus complexe à gérer, car p doit être inséré en mémoire centrale, mais il se peut que celle ci soit pleine vu que p n'a éliminé aucun autre point. Si la mémoire dispose de place, p est ajouté normalement en mémoire centrale. Sinon, p est mis de côté dans le fichier temporaire cité précédemment et sera examiné à nouveau au cours de la prochaine itération de l'algorithme.
  Au vu de ce dernier cas de figure, nous pouvons constater que l'algorithme BNL peut nécessiter un grand nombre d'itérations avant que le skyline final soit calculé.

Cet algorithme fonctionne bien dans le contexte de bases de données de petite ou moyenne taille, car le nombre de points candidats au skyline est réduit. Le fichier temporaire est alors peu utilisé puisqu'il reste pratiquement toujours de la place en

mémoire centrale. Dans le meilleur des cas, l'algorithme se termine après une seule itération avec une complexité $O(n)$ (n représentant le nombre de points dans la base de données). Alors que dans le pire cas, sa complexité est quadratique $O(n^2)$. Cependant, l'algorithme BNL reste toujours plus optimal qu'une requête SQL utilisant l'opérateur SKYLINE, car il limite considérablement les entrées/sorties en chargeant en mémoire centrale un ensemble de points.

L'algorithme Divide & Conquer ($D\&C$) L'algorithme Divide & Conquer a été proposé par les auteurs de [BKS01]. Le principe général de cet algorithme consiste à travailler de manière récursive en divisant l'ensemble de données en plusieurs partitions de sorte à ce que chaque partition puisse être stockée en mémoire centrale. Le skyline partiel des points de chaque partition est calculé en utilisant un algorithme de recherche de skyline se basant sur un calcul en mémoire centrale (ex. l'algorithme BNL). Le skyline final est ensuite obtenu en fusionnant tous les skyline partiels. La complexité de cet algorithme est de l'ordre de $O(n.(\log\ n)^{d-2} + n.\log\ n)$ où n représente le cardinal de l'ensemble de données, et d le nombre de dimensions. Ce résultat est meilleur que la complexité en $O(n^2)$ de l'algorithme BNL. Cependant, l'algorithme $D\&C$ n'est efficace que pour les petits ensembles de données. Pour les bases de données volumineuses, le processus de partitionnement nécessite la lecture et l'écriture de l'ensemble de données au moins une fois. Ceci est extrêmement coûteux en E/S et détériore grandement les performances de cet algorithme. Aussi, cet algorithme n'est pas appropriée pour des traitements en ligne, car aucun point skyline ne peut être retourné tant que la phase de partitionnement n'est pas terminée.

Comme souligné dans [BKS01, CGGL03], la performance moyenne de $D\&C$ et BNL se détériore avec l'augmentation du nombre de dimensions du jeu de données considéré, en raison de l'augmentation du nombre de tests de dominance qui sollicitent de manière intensive le processeur. À partir de ce constat, d'autres variantes de l'algorithme BNL ont été proposées par [CGGL05], pour améliorer ses performances.

L'algorithme Sort Filter Skyline (SFS) Proposé par les auteurs de [CGGL05], cet algorithme est une version améliorée de l'algorithme BNL. Il ordonne toutes les données en entrée à l'aide d'une fonction de score (monotone) correspondant aux préférences des utilisateurs (ex. somme des valeurs des dimensions,...etc.). En ordonnant l'ensemble des points, il assure qu'aucun de ces derniers ne peut être dominé par un point ayant un score inférieur au sien, et cela grâce à la propriété de préservation de la relation de dominance avec toute fonction monotone. Par conséquent, chaque point chargé en mémoire centrale peut immédiatement être retourné comme point skyline. Le nombre de passes nécessaires à effectuer sur le disque pour lire les données est alors égal à la taille de l'ensemble skyline final divisé par la taille de la mémoire centrale (en terme de nombre de points pouvant être chargé à la fois). L'algorithme SFS permet donc de retourner les points skyline de manière progressive et diminue le nombre de comparaisons entre les points.

L'algorithme Linear Elimination Sort for Skyline ($LESS$) L'algorithme $LESS$ [GSG05] est une version optimisée de l'algorithme SFS. Il utilise une mémoire tampon de petite taille, appelée *elimination-filter window* lors de l'étape du tri monotone de l'algorithme SFS. Cette mémoire tampon permet de sauvegarder un petit ensemble de points, qui seront utilisés pour un élagage précoce des autres points qu'ils dominent. Comme mentionné dans [GSG05], $LESS$ surpasse constamment les performances de SFS. Cependant, les performances de SFS et $LESS$ se détériorent grandement avec l'augmentation du nombre de dimensions. En effet, plus l'espace de dimensions est grand, plus les algorithmes SFS et $LESS$ consacrent de temps dans la phase de filtre des skyline. Cela est dû au fait qu'un plus grand nombre de points appartient au skyline et que l'ensemble des données doit être balayé au moins une fois après leur tri.

Tous ces algorithmes basés sur le tri (SFS et $LESS$) pâtissent du grand nombre de pré-calculs requis durant la phase de filtre des skyline, vu que chaque point lu doit être comparé avec les points skyline chargés dans la mémoire tampon.

Les algorithmes avec index Les algorithmes mentionnés ci-dessus ne nécessitent pas que les données en entrée soient indexées. Par contre, une autre catégorie d'algorithmes [GSG05, TEO01, DFS02, MPJ07, PTFS05] exploitent des structures d'index afin d'accélérer le calcul des skyline. Un premier algorithme basé sur les structures d'index de type $B\text{-}Tree$ ou $R\text{-}Tree$ a été proposé dans [BKS01]. Ensuite, deux méthodes de traitement progressifs, $Bitmap$ et $Index$, ont été proposées dans [TEO01]. Différents types d'index ont été exploités pour l'optimisation du calcul des skyline. Ces index peuvent être subdivisés en deux catégories : (i) les index classiques couramment utilisés dans les bases de données, tels que les $B\text{-}Tree$ ou bien les index $Bitmap$ (e.g. les algorithmes $Bitmap$ et $Index$) ; (ii) les index spatiaux, introduits dans le contexte des bases de données spatiales, qui permettent d'appréhender le problème du calcul des skyline sous un aspect géométrique en exploitant certaines propriétés pour l'optimisation du temps de calcul des algorithmes (e.g. les algorithmes NN et BBS). Dans ce qui suit, nous détaillons le fonctionnement de ces algorithmes et l'utilisation de ces différents index.

L'algorithme $Bitmap$ L'algorithme $Bitmap$ convertit chaque point p de l'ensemble de données en un vecteur de bits, représentant le nombre de points ayant une plus petite coordonnée que p dans chaque dimension. La longueur du vecteur de bits est déterminé par le nombre de valeurs distinctes sur toutes les dimensions. Les points skyline sont alors obtenus en utilisant uniquement des opérations binaires sur les vecteurs à bits, permettant une optimisation considérable des temps de calcul des skyline. De plus, l'algorithme $Bitmap$ retournent les points skyline au fur et à mesure de leurs calculs (i.e. c'est un algorithme progressif). Cependant, l'algorithme $Bitmap$ présente un certain nombre de désavantages qui limitent considérablement son application. En effet, la consommation mémoire due à la conversion des points en structure $Bitmap$ peut s'avérer prohibitive pour l'algorithme en présence d'un grand nombre de valeurs distinctes sur les dimensions. Autrement dit, l'algorithme n'est efficace qu'en présence

de dimensions à domaine réduit de valeurs. *Bitmap* gère aussi de manière inefficace les mises à jour. En effet, chaque ajout, suppression ou modification d'une dimension ou d'un point implique le recalcul de tous les vecteurs de bits.

L'algorithme *Index* L'algorithme *Index* organise l'ensemble des données en m listes (m étant le nombre de dimensions de l'espace), chaque liste est alors indexée sous forme d'un arbre *B-Tree* et est ordonnée de manière croissante. La i ème liste contient les points p qui vérifient la condition suivante : $p(d_i) = min_{j=1}^{m}\ p(d_j)$ (i.e. le point p appartient à la i ème liste si $p(d_i)$ correspond à la plus petite valeur de p sur l'ensemble de dimensions associé). Les *B-Tree* sont utilisés pour calculer les skyline locaux de chaque groupe de points, ces derniers seront fusionnés avec les autres skyline pour former le skyline final. Un groupe de points pour chaque liste, représente l'ensemble des points ayant la même valeur sur la dimension correspondante. Bien que cette technique permette de retourner rapidement et progressivement les points skyline en haut des listes, l'ordre dans lequel ces points sont extraits est fixé sans prise en compte des préférences définies par l'utilisateur.

L'algorithme Nearest Neighbor (*NN*) L'algorithme *NN* a été la première méthode de calcul de skyline exploitant les index spatiaux. Comme son nom l'indique, l'algorithme *NN* est basé sur des requêtes de plus proche voisinage, utilisant une distance donnée (e.g. distance de Manhattan). Ces requêtes sont implémentées à l'aide d'index de type *R/R∗-Tree*. L'algorithme utilise une constatation géométrique très simple : les points les plus intéressants sont ceux qui sont proches de l'origine du repère. À partir de cette observation, un algorithme de calcul de skyline efficace (i.e. l'algorithme *NN*) a été proposé dans [DFS02]. Cet algorithme, commence en cherchant le point le plus proche de l'origine du repère. Ce dernier représentant forcément un point skyline puisqu'aucun autre point n'a pu le dominer. Ensuite, il segmente le reste des points en partitions (i.e. zones de l'espace) qui se chevauchent en appliquant la technique de recherche du plus proche voisin sur le dernier point extrait. Les prochains plus proches voisins sont ensuite itérativement trouvés dans chaque partition. Lors de la découverte d'un nouveau point, l'algorithme utilise l'observation citée précédemment, pour éliminer les régions de l'espace à ne plus explorer. Ainsi, il suffit de chercher uniquement dans les régions susceptibles de contenir d'autres points candidats. Chaque nouveau point skyline p découvert est à l'origine de m appels récursifs de l'algorithme (m étant le nombre de dimensions de l'espace), une partition étant ajoutée pour chaque coordonnée de p. A chaque itération, l'arbre *R/R∗-Tree* sera parcouru plusieurs fois pour éliminer tous les doublons dans les zones de chevauchement de toutes les partitions (i.e. certaines zones de l'espace peuvent être parcourues plusieurs fois), ce qui engendre des calculs supplémentaires inutiles. Globalement, l'algorithme *NN* est un bon algorithme pour le calcul des skyline à faible dimensions. Cependant, il est difficilement exploitable dès que le nombre de dimensions dépasse 4.

L'algorithme Branch and Bound Skyline (BBS) L'algorithme BBS a été introduit par les auteurs de [PTFS03, PTFS05]. Comme l'algorithme NN, BBS est aussi basé sur la recherche des plus proches voisins. Il part de la racine du $R\text{-}Tree$ et l'explore en descendant seulement dans les branches qui lui sont utiles. C'est pour cela que l'on dit qu'il est optimal en nombre d'E/S [PTFS03]. L'algorithme BBS utilise un tas qui va contenir les entrées (feuilles / noeuds intermédiaires) de l'arbre en question, triée par ordre croissant selon leur distance minimale ($MinDist$) à l'origine du repère. Au début, toutes les entrées au niveau du noeud racine sont insérées dans le tas en utilisant leur $minDist$ comme index de tri. Ensuite, l'algorithme retire l'entrée ayant la plus petite valeur $MinDist$ du tas et insère ses noeuds fils comme entrées dans ce dernier. Lorsque l'entrée courante est un point non dominé, il est directement ajouté au skyline et tous les noeuds/ sous-arbres qu'ils dominent sont définitivement élagués du $R\text{-}Tree$. L'algorithme s'arrête quand le tas est vide.

BBS est l'algorithme qui effectue le moins d'entrées sorties parmi tous les algorithmes utilisants les $R\text{-}Trees$ (incluant NN). Il constitue la méthode de calcul la plus efficace que nous ayons étudiée pour la recherche des skyline. Il est donc le plus appropriée dans le contexte de bases de données volumineuses et à dimensionnalité élevée.

2.2.3.2 Algorithmes de recherche dans des sous-espaces

Comme nous l'avons vu dans la section précédente, de nombreux travaux récents se sont penchés sur le problème de l'extraction des points skyline en proposant des méthodes de calcul efficaces. Cependant, lorsque le nombre de dimensions est trop important, les requêtes skyline commencent à perdre leur pouvoir discriminant en renvoyant une grande partie des données. Cela dit, est-il vraiment pertinent de prendre en considération l'ensemble des dimensions à chaque nouvelle requête skyline ? La réponse est non. En effet, il a été constaté, que suivant leurs centres d'intérêts, les utilisateurs pouvaient être amenés à poser des requêtes sur différents sous-ensembles de l'espace de dimensions et non sur la totalité. À partir de cette observation, divers travaux se sont alors intéressés à l'extraction de points skyline dans des sous-espaces de dimensions. Dans un scénario général, les requêtes skyline classiques peuvent être directement adaptées dans des sous-espaces. Cependant, lorsqu'il s'agit de calculer les skyline sur différents sous-espaces, aucun des algorithmes existants ne sait exploiter à son avantage les liens qui peuvent exister entre les différents skyline de ces différents sous-espaces. De plus la connaissance de ces liens peut être très intéressante non seulement pour économiser les temps de calcul mais aussi pour un décideur qui souhaite comprendre les raisons pour lesquelles un point donné devient dominant ou à l'inverse dominé. Diverses études se sont alors focalisées sur ce problème [PJET05, RPK10, HGSW08, TD06, LOTW06, JTEH07, TXP08]. Dans ce qui suit nous décrivons un ensemble d'algorithmes spécifiquement développés pour l'extraction de points skyline dans des sous-espaces.

L'algorithme Skyline Cube ($SKYCUBE$) L'algorithme $SKYCUBE$ a été indépendamment proposé par les auteurs de [PJET05, YLL+05]. À l'image du cube de données dans les entrepôts de données, le $SKYCUBE$ consiste en l'ensemble des skyline sur

tous les sous-ensembles possibles. En d'autres termes, on peut dire que le $SKYCUBE$ est au skyline ce que le cube de données est au $GROUP\text{-}BY$: une généralisation multidimensionnelle. Le calcul d'un skyline étant généralement aussi coûteux que le calcul d'un $GROUP\text{-}BY$, la structure utilisée par l'algorithme $SKYCUBE$ a les mêmes inconvénients que le cube de données. En effet, si on souhaite répondre rapidement à toutes les requêtes posées sur la structure de $SKYCUBE$, il faut envisager là aussi une pré-matérialisation de ce cube. Deux différentes implémentations du $SKYCUBE$ ont été proposées :

- The Bottom-Up Skycube algorithm (BUS) : L'idée de base de l'algorithme BUS est de calculer chaque cuboïde (i.e un sous ensemble du cube de données) du $SKYCUBE$ niveau par niveau et de bas en haut, en prenant en compte la propriété suivante : un cuboïde père contient l'union de ses fils. Lorsque l'on calcule un cuboïde, on sait donc qu'un point présent dans un des fils sera forcément skyline. Cette propriété a le double avantage de réduire l'entrée pour le calcul d'un cuboïde et de diminuer le nombre de tests de dominance effectués.

- The Top-Down Skycube algorithm (TDS) : À l'inverse de BUS, l'algorithme TDS permet de calculer chaque cuboïde du $SKYCUBE$ niveau par niveau et de haut en bas. Il calcule d'abord le cuboïde de tout l'espace de dimensions, ensuite chaque cuboïde fils peut être obtenu en appliquant un algorithme de calcul de skyline classique sur le cuboïde père. Cela permet de diminuer le nombre de tests de dominance en réduisant l'ensemble en entrée.

Cependant, ces deux algorithmes partagent certains inconvénients. En effet, pour construire un $SKYCUBE$, les deux méthodes doivent énumérer et rechercher les points skyline associés à tous les sous-espaces possibles et non vides (i.e. $2^n - 1$, avec n le nombre de dimensions l'espace). Cela conduit naturellement à de mauvaises performances sur des ensembles de données à dimensionnalité élevée. Par exemple, pour un ensemble de données associé à 30 dimensions, il existe $2^{30} - 1$ sous-espaces non vides.

Pei et al. [JAWCXH07] ont proposé *Stellar*, une méthode de calcul du *SKYCUBE* qui évite de rechercher l'ensemble skyline dans chaque sous-espace. En utilisant les notions de *Groupe Skyline* et de *Sous-espace décisif*, *Stellar* garantit qu'en conservant uniquement les groupes skyline et les sous-espaces décisifs associés, il est possible de retrouver tous les points skyline. Cette représentation est donc sans perte d'information et plus réduite que le $SKYCUBE$.

L'algorithme *Orion* [RPK10] est à notre connaissance l'approche la plus récente pour le calcul et l'optimisation du *SKYCUBE*. Il propose une représentation concise du *SKYCUBE*. Il réduit considérablement les tests de dominance dans un sous-espace donné en identifiant les points skyline qui peuvent être dérivés à partir de skyline d'autres sous-espaces. En effet, l'algorithme identifie deux types de points skyline : (i) des skyline pouvant être entièrement déduits à partir d'une seule règle d'inférence et (ii) des skyline ayant besoin de règles de dérivation avancées ou, éventuellement de tests de dominance. L'algorithme, en se basant sur les connexions de Galois, réduit également le nombre de sous-espaces à explorer. En effet, un opérateur de fermeture ainsi que

son dual ont été définis pour le *SKYCUBE*. Un de ces opérateurs est ensuite utilisé pour construire une représentation concise du *SKYCUBE*. Avec un parcours en profondeur, l'algorithme *Orion* permet de calculer de manière optimale les sous-espaces clos et d'élaguer efficacement l'espace de recherche. Les expérimentations présentées dans [RPK10] montrent que les performances de l'algorithme *Orion* surpasse les performances de *Stellar*. En effet, contrairement à *Orion*, *Stellar* ne tire pas profit de toutes les propriétés ni de toutes les relations entre les points skyline projetées sur différents sous-espaces. Ces relations peuvent aider à déduire directement les skyline sans test de dominance.

L'algorithme $SUBSKY$ Proposé par les auteurs de [YXJ06], l'algorithme $SUBSKY$ permet le calcul de requêtes skyline dans des sous-espaces arbitraires. $SUBSKY$ convertit les données multidimensionnelles en valeurs à une dimension qui sont indexées dans un $B\text{-}Tree$. Deux différentes implémentations de l'algorithme ont été développées : Basic $SUBSKY$ pour les données uniformément distribuées et General $SUBSKY$ pour les données distribuées de manière aléatoire. Les données sont classifiées dans un $B\text{-}Tree$ sur la base de leur distance à un point repère, qui est identifié pour chaque cluster. Étant donné que la puissance d'élagage de cette méthode repose sur le choix des points repères, $SUBSKY$ n'est pas adapté pour les données dynamiques, où la distribution des données peut changer au fil du temps. En outre, la capacité d'élagage de $SUBSKY$ se détériore rapidement lorsque le nombre de dimensions augmente, ce qui le rend inapproprié pour le calcul de skyline dans des sous-espaces arbitraires à forte dimensionnalité.

2.2.4 Discussion et conclusion

Dans la première partie de cette section, nous avons étudié les meilleurs représentants des deux grandes familles d'algorithmes (avec et sans index) de calculs de skyline. Dans ce qui suit, nous présentons un tableau récapitulatif en prenant en compte les critères d'évaluation décrits précédemment. Les algorithmes de calcul de skyline dans des sous-espaces ne sont pas inclus dans cette comparaison, car ces méthodes se basent sur les algorithmes de calculs de skyline dans un espace complet pour extraire leur skyline. En effet, les algorithmes $SUBSKY$ et $SKYCUBE$ calculent leur skyline avec les algorithmes $D\&C$ et BBS respectivement.

Nous constatons (cf. Table 2.3), que tous les algorithmes décrits précédemment respectent parfaitement les deux critères *Correction* et *Complétude*, en retournant le skyline exact. S'agissant du critère *Efficacité*, nous remarquons que les algorithmes utilisant les index spatiaux sont plus efficaces que les autres. Toutefois, ils deviennent peu efficaces voire inexploitables dès que le nombre de dimensions dépasse 6. Incontestablement, le comportement progressif (i.e. critère *Progressivité*) implique un pré-traitement des données en entrées, c'est à dire l'utilisation d'index (NN, BBS, $Index$ et $Bitmap$) ou le tri des données (SFS et $LESS$). Ces pré-traitement peuvent être réutilisés par toutes les requêtes ultérieures de manière dynamique avec la structure correspondante. Concernant la gestion des mises à jour des données, les algorithmes NN et BBS utilisent des structures d'index dynamiques (R-Tree), pouvant être mises à jour de manière incré-

Algorithme	Efficacité	Progressivité	Préférences	Correction	Complétude
BNL	Non	Non	Non	Oui	Oui
$D\&C$	Non	Non	Non	Oui	Oui
SFS	Non	Oui	Non	Oui	Oui
$LESS$	Non	Oui	Non	Oui	Oui
$Bitmap$	Non	Oui	Non	Oui	Oui
$Index$	Non	Oui	Non	Oui	Oui
NN	Oui	Oui	Oui	Oui	Oui
BBS	Oui	Oui	Non	Oui	Oui

TABLE 2.3 – Comparaison des algorithmes de calcul de skyline dans un espace complet

mentale. Par contre, la maintenance des structures utilisées par les algorithmes SFS et $LESS$, peuvent être améliorées à l'aide d'un B-Tree. En revanche, l'algorithme $Bitmap$ n'est adapté qu'à des ensembles de données statiques car une insertion / suppression d'un point peut engendrer la modification de la représentation bitmap de nombreux autres points. Concernant le dernier critère *Préférences*, à l'exception de NN, qui peut intégrer des préférences utilisateurs en personnalisant la distance utilisée dans la fonction de score, aucun des autres algorithmes ne prend en charge ou n'intègre de préférences utilisateurs.

Nous constatons que les deux critères *Préférences* et *Efficacité* ne sont respectés que par peu d'algorithmes. Pourtant, ces deux critères sont indispensables au bon fonctionnement d'un algorithme de calcul de skyline. Dans ce travail nous nous intéressons particulièrement à ces deux critères d'évaluation. En effet, en nous positionnant dans un contexte de bases de données volumineuses, un algorithme qui ne répond pas au critère d'efficacité est tout simplement un algorithme inexploitable, notamment en présence de données à forte dimensionnalité. Cependant, en dépassant un certain nombre de dimensions (plus de 10 dimensions), les requêtes skyline perdent de leur efficacité et deviennent inadaptées. Dans ce cas de figure, les utilisateurs sont donc plus intéressés par poser des requêtes sur un sous-espace de dimensions plutôt que sur la totalité. Afin de répondre à ces besoins, d'autres algorithmes de calcul de skyline ont été proposés permettant l'extraction de skyline dans des sous-espaces. Parmi les approches proposées, nous avons présenté celles qui nous semblaient les plus intéressantes dans le contexte de notre travail. L'idée principale véhiculée par ces algorithmes est de permettre à l'utilisateur, suivant ses centres d'intérêt, de sélectionner un ensemble de dimensions et d'extraire les points skyline associés à ces dimensions de manière efficace et rapide. Deux approches importantes ont été proposées, le $SKYCUBE$ qui représente le cube de tous les skyline de tous les sous-espaces possibles et non vides, et $SUBSKY$ qui permet un calcul efficace de requêtes skyline dans des sous-espaces arbitraires. Cependant, aucun de ces algorithmes ne permet d'ajouter des dimensions en assurant un calcul incrémental des skyline.

Par ailleurs, la plupart des travaux cités plus haut supposent que les valeurs des

différentes dimensions sont toutes numériques (comparables), et donc munies d'une relation d'ordre prédéfinie. Un problème intéressant se pose lorsque les utilisateurs sont confrontés à des dimensions définies selon des valeurs catégoriques (incomparables). Dans ce cas de figure, c'est à l'utilisateur de définir un ordre (i.e. préférence) sur les valeurs de ces dimensions. Pour chaque nouvelle préférence le skyline doit être recalculé. Dans notre étude, nous nous positionnons dans un contexte d'entrepôts de données et donc d'analyse en ligne. L'utilisateur doit donc être en mesure de définir ou de changer ses préférences en ligne. Ainsi, sur certaines dimensions l'ordre et le skyline associé peuvent changer dynamiquement. Dans ces conditions, un calcul en ligne des skyline constitue un véritable challenge en présence de données volumineuses.

La prise en compte d'un contexte où les utilisateurs peuvent exprimer différentes préférences (i.e., différents ordres) sur certaines dimensions a retenu l'attention de travaux récents [WFP+08, WPFW09], mais les solutions proposées sont très consommatrices en terme de stockage d'informations additionnelles afin d'obtenir des temps de réponse raisonnables aux requêtes.

Dans ce travail de thèse, nous proposons la méthode d'extraction de skyline EC^2Sky [BCQ12] permettant un calcul incrémental des skyline tout en assurant une modification en ligne des préférences utilisateurs. EC^2Sky fournit à l'utilisateur un moyen efficace d'exprimer ses préférences et de les changer sans être pénalisé par des temps d'attente trop longs. Les performances sont obtenues en ne stockant que le minimum d'informations requises afin d'autoriser des mises-à-jour simples et rapides. En particulier, nous cherchons à éviter de recalculer l'ensemble des points skyline à chaque fois qu'une préférence est modifiée, ce qui devient trop coûteux dans un contexte de données multidimensionnelles volumineuses.

Outre la gestion et la prise en compte des préférences utilisateurs, l'extraction de points skyline en présence de dimensions hiérarchiques nous intéresse aussi. En effet, qu'en est-il du calcul des skyline lorsque les données sont agrégées et les dimensions hiérarchiques ? L'application des algorithmes définis précédemment peut-elle être étendue à ce type de données, ou alors une nouvelle problématique est-elle apparue ?

Les requêtes skyline classiques permettent l'extraction des points skyline dans un contexte de dimensions à un seul niveau hiérarchique, et ne sont donc pas adaptées à ce types de données. Dans la littérature, quelques récents travaux [AWAA09, AWAEA08, WMZJ07] se sont intéressés à l'application des requêtes skyline sur des données agrégées. Ces travaux sont principalement axés sur l'optimisation de requêtes sollicitant les deux opérateurs *Skyline* et *Group-By*, ce dernier étant un opérateur d'agrégation dans les bases de données. Cependant, ces approches se sont contentées d'exécuter ces deux opérateurs l'un après l'autre de manière isolée (i.e. exécution séquentielle). Cela ne correspond pas à un réel couplage entre ces deux opérateurs, car calculer le skyline avant d'appliquer l'opérateur *Group-By* revient à appliquer le skyline à des points individuels sans prendre en compte les connaissances liées à la structure de leur groupe. De même, un skyline appliqué avant un *Group-By* a seulement accès au résumé des informations calculées par les fonctions d'agrégation.

À l'inverse, l'opérateur *Aggregate Skyline* proposé par les auteurs de [MI13], combine les fonctionnalités des deux opérateurs *Skyline* et *Group-By*. En effet, les auteurs

ont adapté la relation de dominance au concept de groupe afin de parvenir à extraire l'ensemble des groupes non dominés par d'autres groupes. En d'autres termes, cet opérateur permet à l'utilisateur de poser des requêtes skyline sur des groupes de points afin de sélectionner le groupe le plus pertinent.

Par contre, on peut constater qu'aucun de ces travaux ne s'est intéressé à l'extraction de points skyline en présence de dimensions hiérarchiques. Nous pensons qu'il serait pertinent pour un décideur de pouvoir naviguer le long des axes des dimensions hiérarchiques (i.e. spécialisant / généralisant) tout en assurant un calcul en ligne des skyline correspondants. Pour répondre à ces besoins, nous proposons la méthode de calcul de skyline $HSky$ qui permet de simuler l'effet des deux opérateurs OLAP *Drill-Down* et *Roll-Up* sur le calcul des skyline. Nous avons défini de nouvelles propriétés sur les préférences, qui permettent à l'utilisateur de spécialiser ou de généraliser les valeurs sur lesquelles portent ces préférences, tout en assurant un calcul en ligne des skyline associés. Les deux contributions EC^2Sky et $HSky$ feront l'objet d'une étude plus approfondie dans les chapitres 4 et 5.

Deuxième partie

N-Catch : un modèle d'entrepôt de données sur le cycle de l'azote dans un bassin versant

Chapitre 3

Processus de construction de l'entrepôt de données *N-Catch*

3.1 Introduction et motivations

La relation entre agriculture et qualité des eaux a une importance environnementale, économique et sociale, du fait des nombreux enjeux qui lui sont liés (eau potable, eutrophisation, pêche,...). Dans le but d'aider à la gestion de l'eau, des modèles agro-hydrologiques ont été développés [CGG+05, GOAC+09b]. Ils couplent des modèles de cultures représentant le développement des cultures et des modèles de transfert des polluants, nitrates ou pesticides par exemple, dans les bassins versants. Le modèle TNT est un exemple de ce type de modèle, conçu pour la simulation des transferts et des transformations de l'azote dans des bassins versants. De tels modèles génèrent de grandes quantités d'informations qui sont rarement stockées ou analysées. Par exemple, seules les concentrations quotidiennes à l'exutoire du bassin versant ou les émissions annuelles dans le sol, l'eau et l'atmosphère [DGOC02] sont généralement stockées, et ce même si des informations plus détaillées (ex. à l'échelle de la parcelle ou du sous-bassin) sont disponibles. Ainsi, une grande quantité d'information est perdue en sortie du processus de simulation en raison du trop grand volume de données qui devrait être stocké et des difficultés à analyser et à transformer ces données en informations exploitables.

L'objectif de cette thèse est de développer des méthodes d'aide à l'analyse des résultats de simulation obtenus par un modèle tel que TNT. Le but est d'obtenir un outil permettant à des acteurs, gestionnaires de bassins versants ou chargés d'étude, d'analyser des résultats de simulation, simulant des scénarios, afin de comprendre où, quand et comment, se produisent les émissions d'azote. L'idée de base est de s'appuyer sur la demande de l'utilisateur pour analyser les résultats de simulation et, de manière interactive, produire des connaissances l'aidant à mieux comprendre une situation et à mieux évaluer l'impact d'une décision. Répondre à une question de l'utilisateur c'est alors répondre à une requête sur une base rassemblant l'ensemble des résultats de simulations, et fournir des moyens d'analyse, d'exploration et de visualisation de la réponse. Afin de répondre à ces besoins, nous proposons de construire un entrepôt de données

pour l'archivage et l'analyse de ces résultats de simulations. Comme mentionné dans le chapitre 1, la technologie entrepôt-OLAP apparait comme une technologie clef pour les organisations désirant améliorer l'analyse de leurs données et leur système d'aide à la décision. En effet, la vocation de l'OLAP est de réaliser une analyse interactive et multidimensionnelle des données de l'entrepôt. Cette analyse en ligne agrège les données pour pouvoir les explorer et les visualiser. Par conséquent, cette solution nous semble la plus adéquate pour notre problématique.

Dans ce chapitre, nous présentons un modèle d'entrepôt de données, *N-Catch* (*Nitrogen in Catchment data warehouse*) [BCQG11, BGOC$^+$13, BGOC$^+$], développé pour stocker et analyser des données issues du modèle TNT. Nous nous intéressons particulièrement aux résultats de simulation du modèle agro-hydrologique TNT [BDR01, BDR$^+$02]. Ces données de simulations incluent des données en entrées (pratiques agricoles observées ou reconstituées, données météorologiques,...) et des variables de sorties (débits et concentrations en nitrates). Dans un premier temps, nous décrivons dans la section 3.2 le modèle agro-hydrologique TNT et le bassin versant du Yar (côtes d'armor) qui a été choisi comme site d'application. Ensuite, nous décrivons, dans la section 3.3 la méthodologie suivie pour la conception et la construction de l'entrepôt de données agro-environnemental *N-Catch*. Finalement, nous concluons, en section 3.4, sur les spécificités de *N-Catch* et sur l'efficacité de son processus de stockage et d'analyse des données de simulation pour l'aide à la décision.

3.2 Les données

Dans cette section, nous allons décrire le modèle TNT ainsi que le bassin versant modélisé à l'aide de ce modèle. Un bassin versant représente le territoire drainé par une rivière en amont d'un point nommé "Exutoire" (Figure.3.1). Ensuite, nous présentons le protocole de simulation ainsi que les entrées/sorties de ce modèle.

3.2.1 Le modèle TNT

Le modèle agro-hydrologique *TNT (Topography-based Nitrogen Transfer and transformations)* [BDR01, BDR$^+$02] a été développé pour étudier les flux d'azote (N) et de nitrates (N-NO3) dans des bassins versants de petite taille (10-50 km^2). Ce modèle est orienté processus et est spatialement distribué afin d'inclure des interactions spatiales [BDR$^+$02]. Il est basé sur le couplage (i) du modèle $STICS$ [BMR$^+$98] qui simule la croissance des plantes ainsi que les bilan d'eau et d'azote (N) dans le sol ; (ii) d'un modèle hydrologique spatialement distribué adapté aux aquifères superficiels [BDR01] et (iii) un modèle pour simuler la dénitrification hétérotrophe dans les sols [HG00]. Par conséquent, TNT est conçu pour simuler le cycle de l'azote de la parcelle au bassin versant et pour analyser les impacts des pratiques agricoles et de l'occupation des sols sur les flux et le stockage d'azote à différentes échelles spatio-temporelles. Ces dynamiques sont généralement simulées par TNT sur une grille de $25 * 25$ m (le pas de 25 m correspond à la taille de la maille traitée par TNT) au pas de temps journalier. TNT a été développé autant pour des finalités de recherche (processus de dénitrification, effet

de la structure de paysages,...) que pour permettre d'évaluer des mesures d'atténuation environnementales dans un contexte d'aide à la décision [MRM+12].

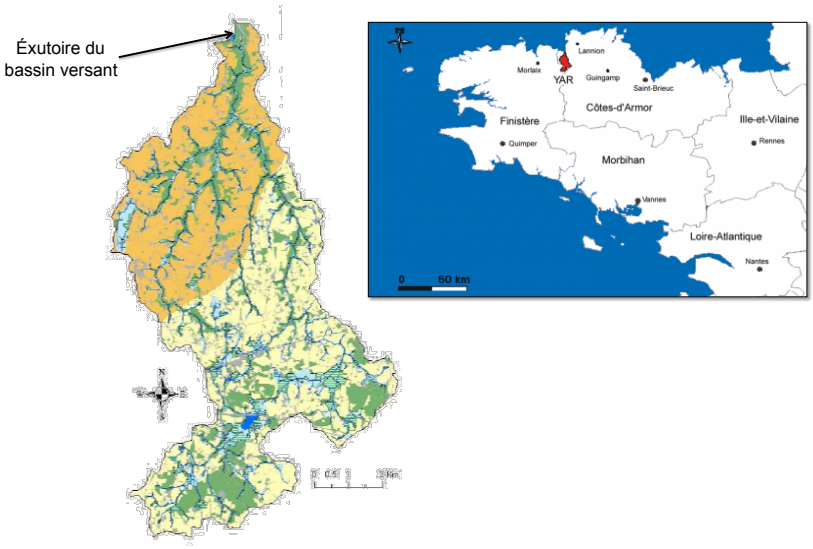


FIGURE 3.1 – Le bassin versant du Yar

3.2.2 Site d'étude

Le modèle TNT a été appliqué sur le bassin versant du Yar (voir Figure.3.1), qui débouche dans la baie Lannion, à l'ouest des Côtes d'Armor. Le Yar draine une surface de $61,5\ km^2$, dont 8% de surfaces urbaines, 28% de forêts et 64% de surfaces agricoles. Il est constitué de 4620 parcelles agricoles. En 2007, 194 fermes avaient tout ou partie de leur surface agricole sur le bassin versant. Malgré des concentrations modérées de l'ordre de 6.8 $mg\ N\text{-}NO3\ l^{-1}$ (i.e. beaucoup plus faible que 11.3 mg N-NO3 l^{-1} qui constitue le seuil à ne pas dépasser pour l'eau potable), la baie de Lannion est affectée par des marées vertes chaque été depuis 40 ans. Une meilleure compréhension du fonctionnement du bassin versant nécessite une analyse détaillée de l'impact des systèmes de culture sur les émissions d'azote à différents échelles spatio-temporelles.

3.2.3 Protocole de simulation

La période de simulation couvre 12 années, du 1 septembre 1996 au 31 août 2008. Des données météorologiques journalières (pluie, ETP, température de l'air minimum et maximum, rayonnement global) ont été acquises auprès de Météo France. Durant cette période, le modèle est calibré à l'aide de données de débits et de concentrations en nitrates ainsi que de nombreuses données sur l'agriculture et le milieu (pratiques agricoles reconstituées, géologie,...) (Figure.3.2). Des scénarios d'évolution des pratiques agricoles sont traduits en données d'entrée, lesquelles sont implémentées dans TNT.

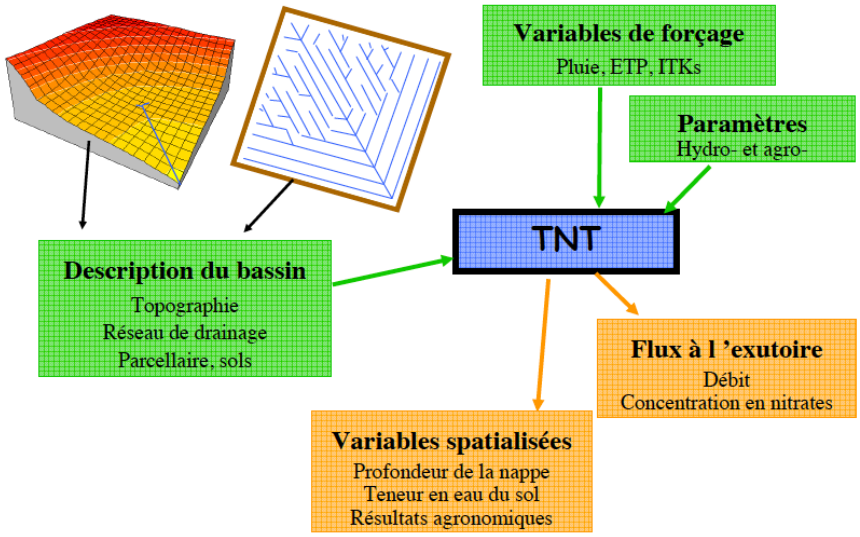


FIGURE 3.2 – Variables d'entrées et de sorties du modèle TNT

Quatre scénarios ont été élaborés pour être testés sur le bassin versant du Yar pour la période 2009-2020 :

- le scénario "pratiques actuelles extrapolées", appelé scénario PAE ;
- le scénario d'application de la Mesure Agro-Environnementale SFEI (Système Fourrager Econome en Intrants, appelé scénario SFEI) ;
- le scénario SFEI auquel on ajoute la mise en prairie permanente des parcelles en zone hydromorphe, appelé scénario SFEI+PP-ZH ;
- le scénario avec la totalité de la SAU implantée en prairie permanente, appelé scénario AA (Arrêt de l'Agriculture) et dont le but est d'évaluer la réactivité du bassin versant.

Les données manipulées ici sont issues de la simulation réalisée sur le scénario PAE. Ce scénario consiste à utiliser les pratiques agricoles définies pour la dernière période de calibrage et à les poursuivre jusqu'en 2020. Les successions culturales, les pressions azotées et l'assolement restent les mêmes.

Les résultats de cette simulation sur 22 ans sont stockés dans deux fichiers distincts (ceci est valable pour tout scénario simulé avec TNT) : (i) le premier représente les rotations des cultures et les opérations correspondantes (semis, fertilisation, récolte) pour chaque parcelle pendant toute la période de simulation, et (ii) le second décrit les teneurs en eau et en azote et les débits à un pas de temps journalier pour chaque parcelle sur l'ensemble de la simulation. Ce dernier fichier peut contenir jusqu'à 44 variables. Sur ces 44 variables, 16 ont été sélectionnées parce qu'elles sont nécessaires au calcul du bilan des émissions d'azote du sol vers la nappe et vers l'air. Ces 16 variables représentent les émissions d'azote vers l'air et la nappe, la dénitrification, la

minéralisation, la profondeur de la nappe, le stock d'azote dans la nappe et le sol, ...etc. (voir liste des variables dans la table 3.1). Ces variables vont nous permettre d'évaluer l'impact des pratiques agricoles sur les principales composantes du cycle de l'azote. Le cycle de l'azote (voir Figure. 3.3) constitue l'ensemble des transformations subies par l'azote (N) dans la biosphère. La plupart sont d'origine microbienne et leur importance est considérable dans la mesure où elles régissent le bilan de l'azote dans le sol, la nappe et l'air et conditionnent la mise à la disposition des formes minérales pour les plantes.

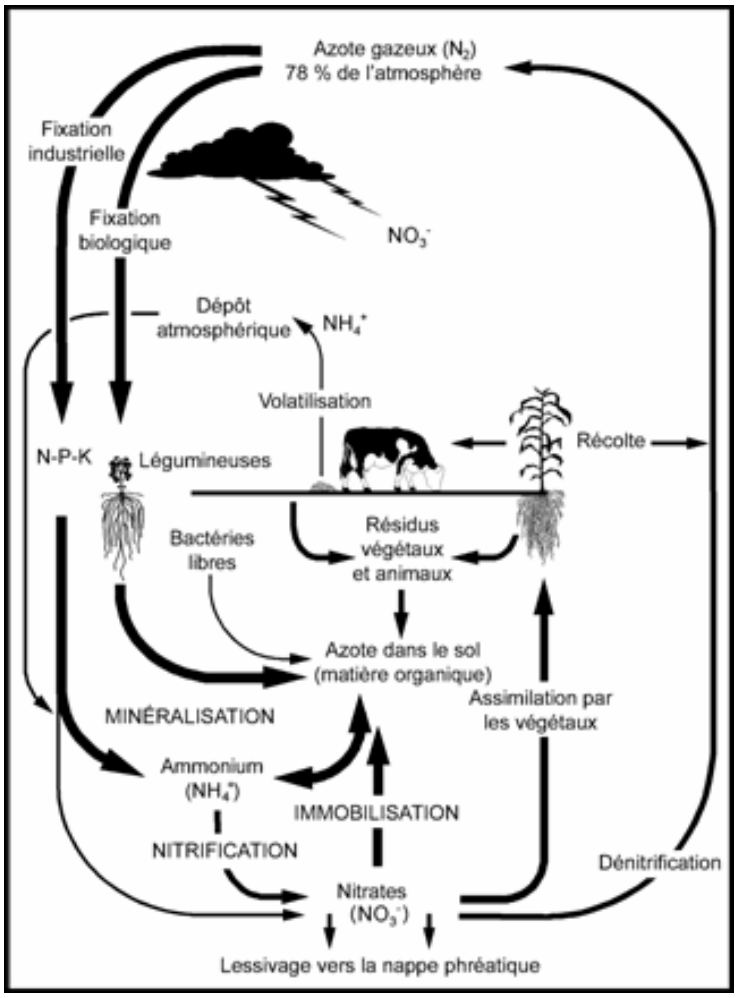


FIGURE 3.3 – Cycle de l'azote

Dans la section suivante, nous allons décrire le pré-traitement de ces données afin de les stocker dans un format multidimensionnel et hiérarchique. Une fois stockées ces données pourront être analysées par les outils OLAP.

Variable	Description
ETR	évaporation-transpiration réelle (mm)
$Nappe$	hauteur d'eau de la nappe (hauteur depuis l'horizon imperméable) (m)
RU	réserve utile en eau d'un sol (quantité d'eau que le sol peut stocker et restituer à la plante, réserve facilement accessible par la plante sur l'ensemble du profil de sol) (mm)
NRU	quantité d'azote stockée dans la réserve utile du sol (g/m^2)
$N\text{-}micro\text{-}alterite$	quantité d'azote stockée dans la microporosité de l'altérite (g/m^2)
$Nnappe$	quantité d'azote dans la nappe (g/m^2)
$ONfix$	quantité d'azote atmosphérique fixée par la plante (g/m^2)
$Nmine$	quantité d'azote minéral résultant de la minéralisation (g/m^2)
$Ndenit$	quantité d'azote minéral perdue par dénitrification (g/m^2)
$Chum$	quantité de Carbone dans l'humus actif (g/m^2)
$N\text{-}sequestre$	quantité d'azote stockée par une matière organique (l'azote mis dans un humus non facilement minéralisant : il ne sera minéralisé que si il y a retournement de prairie) (g/m^2)
$ONplante$	quantité d'azote exportée et fixée par la plante (g/m^2)
$NHum$	quantité d'azote dans l'humus actif (g/m^2)
$NRuissel$	quantité d'azote sortant par ruissellement (g/m^2)
$NExfiltra$	quantité d'azote exfiltrée (g/m^2)
$NEcoulp$	flux d'azote qui part en profondeur (g/m^2)

TABLE 3.1 – Description des variables de sorties TNT sélectionnées

3.3 Processus de construction de N-$Catch$

Les modèles agro-hydrologiques comme TNT génèrent beaucoup de résultats intermédiaires, mais seulement quelques variables peuvent être stockées (principalement des concentrations, des charges et des flux journaliers ou annuels, simulées à l'exutoire du bassin versant). Donc, les variables intermédiaires à différentes échelles spatio-temporelles ne sont pas complètement exploitées. Pourtant, les experts ont souvent besoin de ces résultats intermédiaires pour affiner leurs analyses. Ainsi, un outil est nécessaire pour stocker et manipuler ce gros volume de données de manière efficace. Nous présentons dans cette section notre application N-$Catch$ qui illustre comment un entrepôt de données peut être utilisé de manière innovante pour stocker et analyser efficacement les données intermédiaires simulées par ces modèles intégrés.

3.3.1 Modélisation et alimentation de N-$Catch$

La mise en oeuvre d'un entrepôt de données nécessite un important travail d'étude de l'existant et de recueil de données à partir de sources hétérogènes et diverses pour

bien prendre en compte les objectifs d'analyse. L'entrepôt de données est ainsi conçu pour répondre à un ensemble de besoins d'analyse recensés auprès des utilisateurs. Le processus de construction d'un entrepôt de données se déroule généralement en trois étapes :

- La modélisation multidimensionnelle des données (cf. section 1.2.1) : dans cette phase les données sont organisées selon un schéma multidimensionnel afin de proposer un accès aux données intuitif et très performant ;
- L'alimentation de l'entrepôt : c'est une phase essentielle dans le processus d'entreposage. Elle se déroule en plusieurs étapes : extraction, transformation et chargement des données, qui sont prises en charge par le processus d'*ETL (Extracting, Transforming and Loading)* ;
- Stockage physique des données : cette phase assure le stockage et la gestion des données multidimensionnelles.

Dans ce qui suit nous allons décrire le processus de construction de l'entrepôt de données *N-Catch* en détaillant ces trois phases.

3.3.2 Modélisation de *N-Catch*

La modélisation d'un entrepôt de données commence par la sélection, à partir des données collectées, des dimensions et des mesures qui répondent aux besoins d'analyse des utilisateurs. L'objectif principal de notre étude est de parvenir à quantifier l'impact des pratiques agricoles sur la pollution nitrique à différents niveaux spatio-temporels. À partir de ce besoin et des données de simulations collectées, nous avons identifié trois dimensions : spatiale (Localisation), temporelle (Date) et Pratique Agricole (et 18 mesures dont 16 correspondent aux variables de sortie du modèle TNT et 2 correspondent à des bilans d'émissions d'azote du sol vers la nappe et du sol vers l'air ont été sélectionnées à partir des 16 variables de sorties.

3.3.2.1 Dimension spatiale

La dimension spatiale (i.e. Localisation) est utile pour quantifier l'effet de facteurs environnementaux. Dans l'entrepôt de données *N-Catch*, la dimension Localisation est structurée en deux niveaux hiérarchiques : *parcelle* et *bassin versant* (voir Figure.3.4a). Les entrées/sorties du modèle TNT sont disponibles à l'échelle de la parcelle et de l'ensemble du bassin versant. Grâce à cette dimension, toutes les données (quantitatives) décrites à l'échelle de la parcelle peuvent être agrégées à l'échelle de l'ensemble du bassin versant. Cependant, nous regrettons la non disponibilité, à ce moment là, de plus d'information (attributs) spatiales (sol, exploitation, sous-bassin,...).

3.3.2.2 Dimension temporelle

La dimension temporelle (i.e. Date) est utile pour permettre aux utilisateurs d'analyser les changements ou les effets d'un enchainement d'activités agricoles. Dans l'entrepôt

de données *N-Catch*, la dimension date est structuré en quatre niveaux hiérarchiques : *jour*, *mois*, *année* et *période de la simulation* (voir Figure.3.4b). Les résultats de simulation sont disponibles à un pas de temps journalier, et peuvent, grâce à la structure hiérarchique de la dimension Date, être agrégées à l'échelle du mois, de l'année ou de toute la période de la simulation.

3.3.2.3 Dimension agricole

La dimension Pratique Agricole représente un élément clé de l'entrepôt de données *N-Catch*, puisque le but principal des simulations est de parvenir à expliquer l'impact des pratiques agricoles sur la pollution nitrique dans les cours d'eau. Chaque simulation fournit, pour chaque parcelle agricole, les dates des opérations culturales (semis, récolte, labour et fertilisation) ainsi que les types et les quantités d'apports organiques appliqués, et les rendements des cultures. Nous avons structuré la dimension Pratique Agricole en quatre niveaux hiérarchiques : *Opération culturale*, *Itinéraire technique (ITK)*, *Culture* et *Rotation culturale* (voir Figure.3.4c).

Une opération culturale (ex. semis, récolte et apports minéraux et organiques) a une position chronologique dans l'itinéraire technique dans lequel elle apparaît. Un itinéraire technique [SMDF$^+$12] décrit des opérations culturales successives d'un type de culture donné, tandis que type culture décrit la nature de la culture. Les itinéraires techniques n'étaient pas directement disponibles à partir des données d'entrées de TNT et ont donc dû être calculés. De plus, il a été nécessaire de reconstruire les rotations de cultures à partir des données de dates et de types de culture associées aux 4620 parcelles du bassin versant. Les rotations de cultures et les itinéraires techniques ont été décrits par des séquences temporelles, permettant de gérer (i) les relations temporelles entre les opérations culturales et les itinéraires techniques et entre les types de cultures et les rotations de cultures, et (ii) les occurrences multiples d'un même élément dans une même séquence (ex. l'opération "fertilisation" peut apparaître à deux dates différentes dans un itinéraire technique).

Exemple 8 *Dans cet exemple nous présentons une instance de la dimension Pratique Agricole, l'opération culturale* Op_56 *représentant une* $Fertilisation$ *apparaît dans l'itinéraire technique* $RGA\_minPFP0$ *décrit par la séquence <pâturage, fertilisation, fertilisation, fauchage>. Cet ITK appartient au type culture* Cul_4 *représentant le type Ray-grass Anglais (prairie-RGA) de la rotation culturale* Rot_23 *décrite par la séquence <maïs, prairie-RGA, blé> (Figure.3.5).*

3.3.2.4 Mesures : indicateurs agro-environnementaux

Pour étudier l'impact des pratiques agricoles sur les émissions d'azote, les 16 variables de sorties du modèle TNT, représentant des débits et des concentrations en nitrates, ainsi que les deux bilans de flux d'azote (flux sol-nappe et flux sol-air) constituent les mesures de la table de fait de *N-Catch*. Ces mesures sont associées aux fonctions d'agrégation *somme* ou *moyenne*.

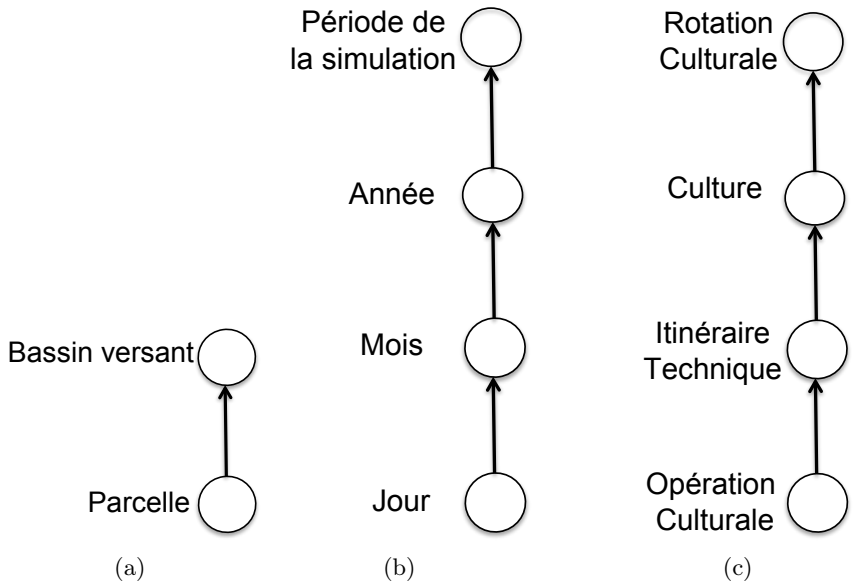


FIGURE 3.4 – Structures hiérarchiques (a) de la dimension spatiale, (b) de la dimension temporelle, (c) de la dimension agricole.

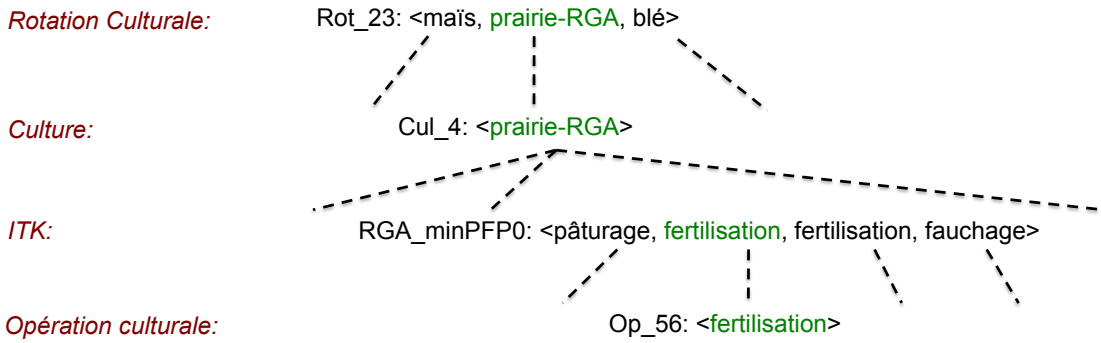


FIGURE 3.5 – Exemple d'une instance de la dimension Pratique Agricole

Une fois les mesures et les dimensions identifiées, nous pouvons choisir le schéma multidimensionnel correspondant. La dimension Pratique agricole décrite précédemment a une structure hiérarchique assez complexe et les liens hiérarchiques ont été difficiles à modéliser. Chaque niveau hiérarchique de cette dimension est décrit par un ensemble d'attributs propres. Pour cette raison, nous avons choisi de combiner le schéma en étoile et le schéma en flocon (i.e. schéma mixte), décrits dans la sous-section 1.2.4, pour la modélisation de l'entrepôt de données *N-Catch*. En effet, les dimensions Date et Localisation ont été modélisées selon le schéma en étoile étant donné qu'elles ne contiennent pas beaucoup d'attributs. Par contre, la dimension Pratique agricole a été modélisée selon le schéma en flocon (i.e. la table Pratique Agricole a été éclatée en plusieurs tables)

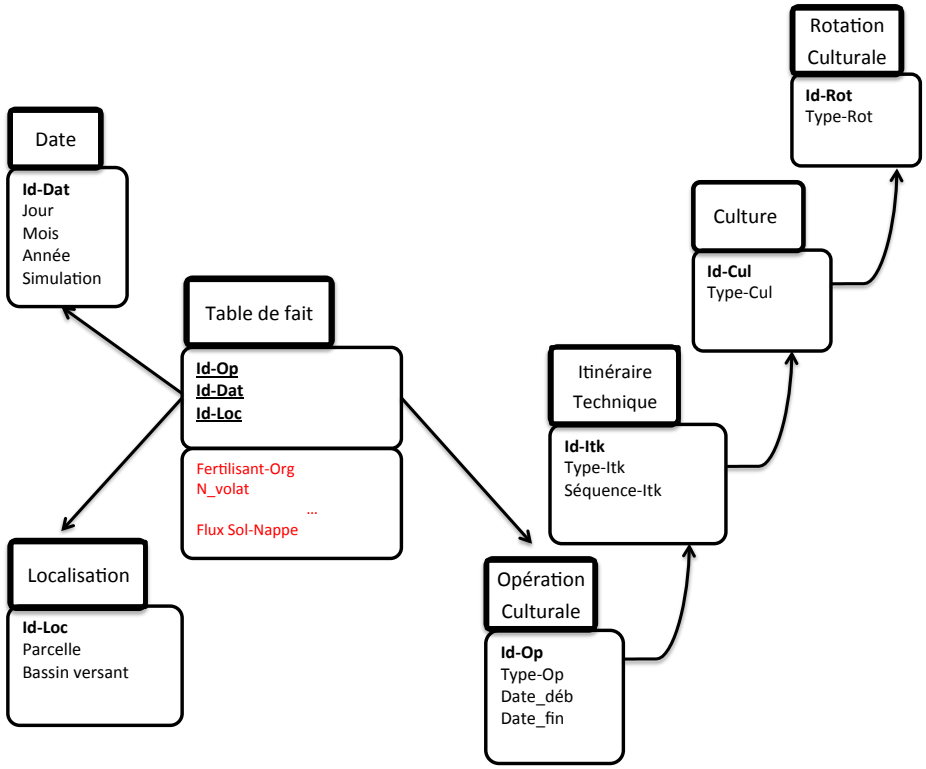


FIGURE 3.6 – Le modèle multidimensionnel de l'entrepôt de données *N-Catch*

pour les raisons citées précédemment. Cette modélisation mixte nous permet (i) de limiter les jointures nécessaires pour l'évaluation d'une requête, grâce à la dénormalisation des tables de dimensions dans le schéma en étoile, et (ii) d'économiser l'espace occupé par les données, grâce à la normalisation des tables de dimensions les plus volumineuses (i.e. éclater les tables de dimensions afin de permettre une représentation plus explicite de la hiérarchie) dans le schéma en flocon, permettant de réduire la redondance de données.

La Figure.3.6 décrit le modèle en flocon à 3 dimensions (Localisation, Date et Pratique Agricole) avec les mesures sélectionnées pour l'entrepôt de données *N-Catch*.

3.3.3 Alimentation de *N-Catch*

Une fois les besoins identifiés, les données collectées et la modélisation multidimensionnelle achevée, la phase *ETL* peut être mise en oeuvre.

3.3.3.1 Extraction des données

Les informations relatives aux trois dimensions ainsi qu'à toutes les mesures sélectionnées sont extraites à partir des entrées/sorties du modèle *TNT*. Les mesures sont extraites des fichiers de sorties du modèle, et les données décrivant les dimensions et

les niveaux hiérarchiques associés sont extraites des données d'entrées. Les fichiers d'entrées sont des documents textuels non structurés qui sont, de ce fait, difficiles à exploiter. Nous avons utilisé des expressions régulières pour analyser (i.e. analyse syntaxique) et extraire les informations qui nous intéressent de ces fichiers. Pour ce faire, des scripts écrits dans le langage de programmation *Perl* (voir site web : http ://www.perl.org/) ont été utilisés. Les fichiers de sorties, quant à eux, sont des documents .csv et sont donc faciles à traiter.

3.3.3.2 Transformation et chargement des données

Une fois les données extraites, il faut alors les pré-traiter pour les homogénéiser. La transformation des données extraites a nécessité des pré-traitement assez importants étant donné que les séquences d'opérations culturales (resp. les séquences de cultures) n'étaient pas directement disponibles. Par conséquent, il a tout d'abord fallu extraire de chaque parcelle, les différents types d'opérations culturales (resp. différents types de cultures) et leurs dates respectives, puis construire les séquences d'opérations culturales (i.e ITK) (resp. les séquences de cultures (i.e. rotations de culture)) afin de les intégrer ultérieurement dans l'entrepôt de données *N-Catch*. D'autres types de transformations ont été réalisées sur ces données brutes : décodage, nettoyage, normalisation/dénormalisation et homogénéisation. Finalement, la dernière étape consiste à charger et à intégrer les données extraites et transformées dans l'entrepôt de données.

3.3.4 Implémentation de *N-Catch*

Notre choix s'est porté sur l'implémentation *R-OLAP* (cf. sous-section 1.2.4.1) et cela pour plusieurs raisons : (i) le modèle multidimensionnel de *N-Catch* (i.e. schéma mixte) est implémenté par l'architecture *R-OLAP* ; (ii) les systèmes basés sur une implémentation *R-OLAP* peuvent stocker de grands volumes de données, et ont la capacité de répondre aux attentes de flexibilité et d'accessibilité des données relationnelles stockées dans la structure multidimensionnelle.

Les résultats de simulations issus du modèle *TNT* représentent un gros volume de données. En effet, pour la simulation considérée [MRR$^+$12] pour l'illustration de *N-Catch*, 16 variables de sorties par parcelle, sur 4620 parcelles et sur 4380 jours, ont été stockées, ce qui correspond à 20.235.600 enregistrements et 9 *Go*. *R-OLAP* est par conséquent adapté à notre problématique.

Si l'on considère une implémentation en relationnel (*R-OLAP*), les faits seront dans une table relationnelle (table de faits) et chacune des dimensions sera dans une table relationnelle (tables de dimension). Pour une exploitation optimale de l'entrepôt de données *N-Catch*, nous avons utilisé des index (de type *B-Tree*) sur les colonnes les plus fréquemment interrogées afin de simplifier et d'accélérer les opérations de recherche, de tri, de jointure ou d'agrégation effectuées par le *SGBDR*. Nous avons aussi utilisé les vues matérialisées pour améliorer les performances des requêtes. Une vue est une table contenant les résultats d'une requête. Ainsi, nous avons matérialisé les vues des requêtes les plus fréquentes, ou pouvant être partagées par plusieurs requêtes, au lieu

de matérialiser toutes les vues possibles. En effet, lors de la phase d'identification de besoins, nous avons remarqué que les décideurs étaient plus intéressés par certains niveaux hiérarchiques des différentes dimensions que par d'autres. Par exemple, pour la dimension Pratique Agricole (resp. Localisation), le niveau hiérarchique Culture (resp. Parcelle) apparaît dans beaucoup de requêtes. Aussi, dans les types de cultures, seules les cultures les plus représentées (7 cultures dans notre cas) dans le bassin intéressaient les experts. À partir de ce constat, nous avons décidé de matérialiser les agrégats relatifs aux types de cultures les plus représentés à l'échelle de la parcelle et de l'année. Ces vues matérialisées peuvent être utilisées pour répondre à d'autres requêtes. Par exemple, des requêtes portant sur ces mêmes cultures mais agrégées à des échelles spatio-temporelles plus générales que l'échelle de la parcelle ou de l'année respectivement.

3.4 Utilisation de l'entrepôt de données *N-Catch*

L'entrepôt de données *N-Catch* est utilisé pour analyser de manière approfondie, dans l'espace et dans le temps, les effets des pratiques agricoles sur les émissions d'azote dans l'eau et dans l'air. Les avantages de la structure multidimensionnelle de *N-Catch* sont illustrés en considérant chacune de ses dimensions, et en se basant sur les données simulées durant une période de 4380 jours (i.e. 12 premières années).

Dans un premier temps, pour visualiser les résultats des requêtes *OLAP* sous différentes formes graphiques, nous avons utilisé le moteur *OLAP* Mondrian de la plateforme décisionnelle Pentaho (Open Source) (voir site web : http ://mondrian.pentaho.com/) qui permet, à partir d'un entrepôt de données stocké dans un système de gestion de bases de données relationnelles, de construire des cubes de données, qui peuvent être ensuite interfacés avec des applications de visualisation (telles que JPivot, Pentaho Analyzer, Pentaho Analysis Tool, Geo Analysis Tool, etc.). L'inconvénient de cet outil est que sa version gratuite est adaptée à la gestion et à l'affichage de petits volumes de données. En effet, dès que le volume de données augmente, l'affichage devient long et illisible, donc inexploitable. Nous avons donc décidé d'utiliser le logiciel *R* (voir site web : http ://cran.r-project.org/) pour transformer les résultats de nos requêtes en graphiques. *R* est un logiciel de calcul scientifique interactif libre qui possède une large collection d'outils statistiques et graphiques. Avec cet outil, la restitution graphique de gros volumes de données est de bonne qualité. Dans la suite de cette section, tous les graphiques présentés sont réalisés avec *R*.

3.4.1 Requêtes étudiées

Après plusieurs échanges avec les décideurs et les futurs utilisateurs de *N-Catch* (comité de la Lieu de Grève et chercheurs en agro-hydrologie de l'UMR SAS/INRA Rennes), nous avons collecté un ensemble de questions couvrant bien les enjeux du territoire, et auxquelles l'entrepôt de données *N-Catch* pouvait apporté des réponses. Ces questions portent sur des interactions spatio-temporelles, des interactions milieu-pratiques agricoles ou sur l'usage des sols et d'assolement. Les questions qui nous ont parues possibles à traiter sont listées ci-dessous :

1. Quelles parcelles et quel(s) système(s) de production génèrent le plus de fuites d'azote ? Le but de cette requête est de parvenir à caractériser les parcelles (selon leur type de sol, leur position topographique dans le bassin versant,...etc.) et les systèmes de cultures (selon les types de cultures, la durée des rotations culturales,...etc.) les plus polluantes du bassin versant.
2. Quelles pratiques agricoles sont optimales sur les zones humides ? Les zones humides constituent des zones de dénitrification, des zones d'intérêts écologiques. Il est donc important d'identifier les pratiques agricoles les plus adaptées (i.e. pratiques agricoles avec un faible niveau de fertilisation,...etc.) à ces milieux.
3. Combien de temps faut-il faire durer les prairies (5, 7, 10 ans) pour limiter les fuites d'azote à différentes échelles spatiales ? L'objectif de cette question est de parvenir à identifier la durée optimale d'une prairie temporaire dans une succession culturale, c'est-à-dire pour laquelle les fuites d'azote sont les plus basses.

À partir des données de simulations stockées, ces questions sont du ressort de l'entrepôt de données *N-Catch*. Chaque question peut se traduire par un ensemble de requêtes *OLAP* classiques. Dans ce qui suit, nous illustrons l'utilisation de l'entrepôt de données sur la question n^o1. Nous avons choisi de détailler le traitement de cette question, car elle porte sur les trois dimensions de *N-Catch*. Ceci nous permet d'illustrer la navigation sur les trois dimensions avec la même requête.

3.4.2 Illustration de *N-Catch*

Dans cette sous-section, nous souhaitons identifier les parcelles et les systèmes de cultures les plus (resp. les moins) polluants. Pour ce faire, nous allons montrer comment exploiter les différentes dimensions de *N-Catch* et leurs différents niveaux hiérarchiques pour retrouver, au niveau de granularité souhaité, des éléments de réponses à la requête de l'utilisateur. La pollution nitrique sera quantifiée grâce aux deux mesures de *N-Catch* : les deux bilans de flux d'azote (flux sol-nappe et flux sol-air). Un accent sera mis sur le bilan flux sol-nappe, étant donné que notre étude s'intéresse principalement à la qualité de l'eau.

Les graphiques présentés dans cette sous-section ont été obtenus après plusieurs interactions avec les utilisateurs (plusieurs experts ont contribué à l'obtention de ces graphiques). En effet, les utilisateur soumettent leur requête (i.e. question n^o1) une première fois à l'entrepôt *N-Catch* en sélectionnant des mesures et un ensemble de dimensions avec le niveau de granularité souhaité avec des contraintes de filtrage. Puis ils affinent leur requête interactivement en ajoutant ou en supprimant des niveaux de hiérarchie ($Roll-up$ et *Drill-down*), en ajoutant ou modifiant des conditions de filtrage (*Slice*) ou enfin en déplaçant des colonnes d'un axe à l'autre (*Dice*). Ils peuvent aussi changer et améliorer la présentation de ces résultats en utilisant différents types de graphiques, afin de faciliter l'interprétation des résultats. Par exemple, au départ les utilisateurs avaient souhaité afficher les dynamiques agricoles du flux d'azote du sol vers la nappe à l'échelle de l'opération culturale. Une fois ces résultats affichés, ils ont conclu qu'aucune information pertinente ne pouvait être extraite à cette échelle. Il ont donc

affiné leur requête en généralisant les résultats à l'échelle de l'ITK et du Type Culture. Ces résultats ont été affichés sous forme de nuages de points. Les utilisateurs ont pu extraire des informations intéressantes à partir de ces graphes, et identifier les ITK et les cultures qui émettaient le plus (resp. le moins) d'azote dans la nappe à l'échelle du bassin versant. Cependant, les utilisateurs ont regretté le fait de ne pas pouvoir extraire d'informations sur la variabilité intra-culture avec la visualisation en nuages de points. Ils ont donc demandé à changer de type de graphique d'affichage et opté pour des boites à moustaches (voir Figure.4.3). Effectivement, l'information recherchée était beaucoup plus claire et accessible avec ce nouveau type de graphique.

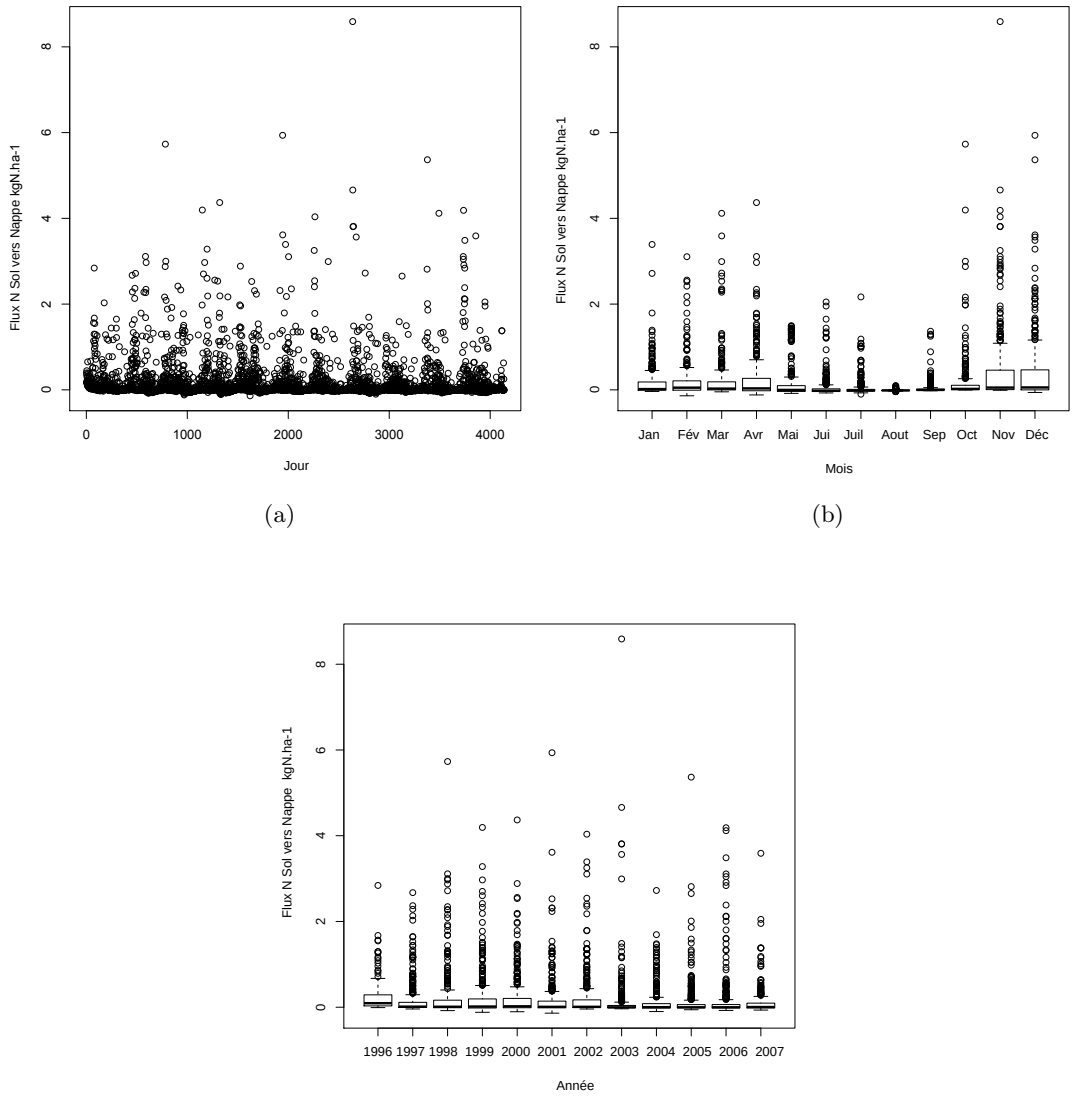
Une fois les résultats validés, les utilisateurs ont proposé une interprétation de ces derniers. Dans ce qui suit nous présentons ces résultats ainsi que l'interprétation thématique des experts.

3.4.2.1 Dimension temporelle

La dimension temporelle facilite l'agrégation des données à différents pas de temps, ainsi que l'analyse des dynamiques temporelles. La dynamique des flux journaliers d'azote du sol vers la nappe montrent une répétition de motifs saisonniers, avec des quelques valeurs remarquables. Au pas de temps mensuel, des différences saisonnières apparaissent clairement, avec une augmentation nette en automne et des valeurs remarquables plus fréquentes durant la période de réhumectation du bassin versant (Figure.4.1). À un pas de temps annuel, des valeurs plus élevées surviennent pendant le début de la période de simulation. Des différences apparaissent selon les années, avec des flux d'azote plus élevés entre 1997-1999 et entre 2005-2006 qui s'expliquent par des volumes annuels de précipitations plus élevés.

3.4.2.2 Dimension spatiale

À l'échelle de la parcelle, l'entrepôt de données *N-Catch* peut être utilisé pour analyser des relations entre des mesures sur des types d'occupation du sol différents, par exemple, entre celles des prairies permanentes (Prairie-P) et des prairies temporaires (Prairie-T) (Figure.4.2). Une prairie permanente est une surface consacrée à la production d'herbe et d'autres plantes fourragères herbacées depuis au moins 5 ans. Une prairie temporaire, quant à elle, est une prairie renouvelée depuis moins de 5 ans. Dans cet exemple, la fonction d'agrégation utilisée est la somme des valeurs de la mesure associée dans le temps, divisée par la durée de la rotation, pour obtenir une valeur moyenne par an. Une forte corrélation positive ($r = 0.8$) (Figure.4.2a) existe entre le flux d'azote du sol vers l'air et la dénitrification, tandis que, pour la corrélation entre le flux d'azote du sol vers la nappe et la minéralisation est faible (Figure.4.2b). Cependant, on remarque que les parcelles sont dispersées en deux groupes en fonction de l'occupation des sols (Figure.4.2). Pour la même quantité d'azote minéralisé, le flux d'azote du sol vers la nappe est beaucoup plus faible dans les prairies permanentes que dans les prairies temporaires. Une corrélation positive ($r = 0.3$) faible est observée entre le flux d'azote du sol vers la nappe et la minéralisation (Figure.4.2b). Cependant, plus la



(a) (b)

(c)

FIGURE 3.7 – Dynamiques temporelles simulées du flux d'azote (N) du Sol vers la Nappe sur une période de 12 années : (a) distribution par jour, (b) distribution par mois, et (c) distribution par année.

quantité de minéralisation est élevée, plus le flux d'azote du sol vers la nappe varie. Il n'existe aucune corrélation entre la fertilisation et le flux d'azote du sol vers la nappe, mais au-dessus d'une certaine valeur de la quantité d'azote apporté par fertilisation ($\approx 76\ kgN.ha^{-1}.y^{-1}$), la variabilité des flux d'azote du sol vers la nappe augmente (Fi-

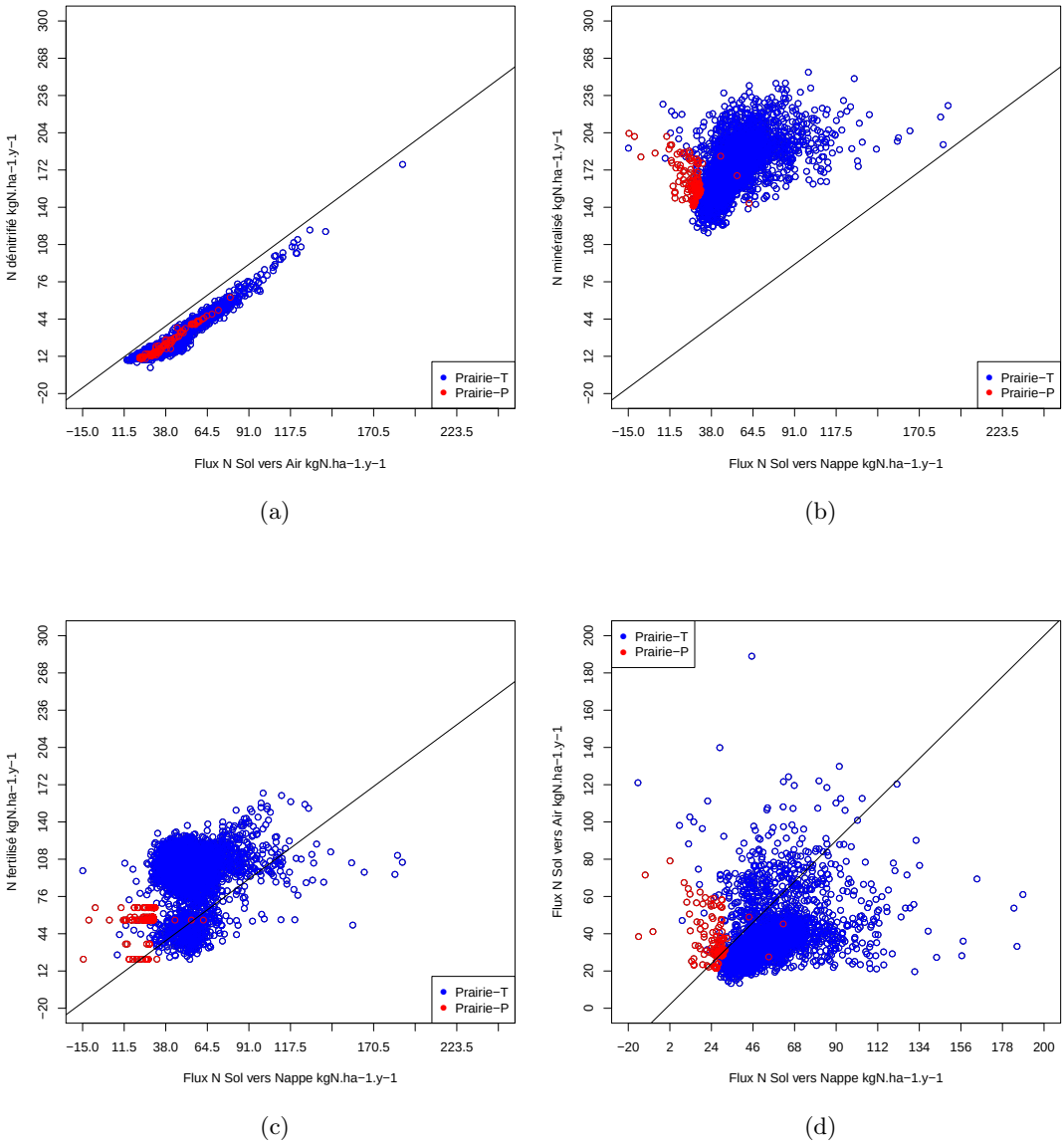
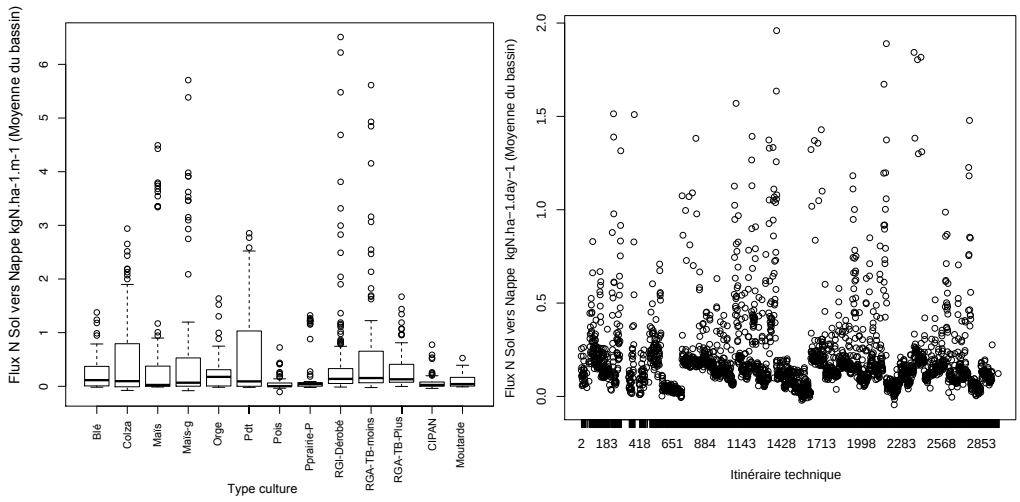


FIGURE 3.8 – Analyse de processus à l'échelle de la parcelle : (a) N dénitrifié vs. Flux N Sol vers Air, (b) N minéralisé vs. Flux N Sol vers Nappe, (c) N fertilisé vs. Flux N Sol vers Nappe, et (d) Flux N Sol vers Nappe vs. Flux N Sol vers Air



(a) (b)

FIGURE 3.9 – Dynamiques agricoles du flux N du Sol vers la Nappe sur une période de 12 années :(a) moyenne du bassin versant par type de culture et par mois et (b) par itinéraire technique

gure.4.2c). Pour la corrélation entre le flux d'azote du sol vers la nappe et celui vers l'air, les parcelles sont de nouveau dispersées en deux groupes en fonction de l'occupation des sols (Figure.4.2d). Le flux d'azote du sol vers la nappe est beaucoup plus faible pour le même flux d'azote du sol vers l'air dans les prairies permanentes que dans les prairies temporaires.

3.4.2.3 Dimension agricole

En considérant toutes les parcelles du bassin versant, l'analyse des variations quotidiennes du flux d'azote du sol vers la nappe révèle des relations complexes entre les pratiques agricoles (Figure.4.3). La fonction d'agrégation utilisée dans cet exemple correspond à la moyenne des flux journaliers d'azote du sol vers la nappe par mois et à l'échelle du bassin versant. Les différences de flux d'azote entre les cultures existent, mais demeurent faibles (ex : la différence du flux d'azote du sol vers la nappe de la culture blé et celui de la culture maïs est faible). Cependant, la variabilité intra-culture est étonnamment élevée. Par exemple, une grande partie des flux mensuels d'azote du sol vers la nappe de la culture pommes de terre varient fortement. Cette variabilité peut être due au climat, à la position topographique dans le bassin versant ou à la façon dont la culture s'insère dans une rotation. Les effets des pratiques agricoles peuvent être observés dans les variations de la moyenne des flux journaliers d'azote du sol vers la nappe

1. RGA-TB-Moins et RGA-TB-Plus dans la Figure.4.3a signifie Ray-Grass Anglais-Trèfle Blanc avec moins (resp. plus) de 40% de trèfles dans la biomasse aérienne.

par itinéraire technique, qui intègre le type de culture et les opérations culturales, au niveau du bassin versant (Figure.4.3b). Les différences sont beaucoup plus faibles entre les itinéraires techniques qu'entre les types de cultures, ce qui indique que les cultures doivent être considérées dans leur mode de gestion. Le type d'itinéraire technique influence la présence ou non de valeurs remarquables.

3.4.3 Discussion

À partir des exemples précédents, nous constatons que l'analyse en ligne rend visible des informations cachées dans les données emmagasinées dans l'entrepôt *N-Catch*. Cependant, nous remarquons que les données ne sont pas toujours exploitées selon leur plein potentiel et une partie de leur richesse, c'est-à-dire, leur composante spatiale, lorsqu'elle est disponible, est souvent inutilisée. En effet, dans les exemples précédents, grâce à l'analyse des parcelles sous forme de nuages de points, nous avons pu identifier les parcelles les plus polluantes dans le bassin versant du Yar. Cependant, avec ce type de graphiques, seuls les numéros de parcelles étaient fournis comme information. Or l'utilisateur souhaiterait pouvoir visualiser ces parcelles (i.e. la référence spatiale de ces parcelles) sur une carte afin de connaître, par exemple, leur position dans le bassin versant (i.e. bas de bassin, haut de bassin,...), l'étendue de la pollution nitrique sur le bassin versant, etc. En effet, une simple visualisation de cette composante spatiale permet de mieux comprendre le phénomène en question (i.e. pollution nitrique) en voyant sa position dans un cadre de référence géographique. Souvent, l'affichage cartographique révèle des informations (ex. proximité entre deux phénomènes isolés, étendue d'un phénomène, forme d'un phénomène longeant une rive, ...) qui n'auraient pas été soupçonnées en faisant appel à d'autres méthodes de représentation. Ainsi, l'utilisateur a exprimé le besoin de pouvoir examiner les détails d'une région d'intérêt (ex : parcelle) pour les comparer avec ceux d'une autre région qui n'est pas adjacente, et analyser s'il y a des caractéristiques communes entre les distributions spatiales du phénomène dans ces deux régions. L'utilisation de la carte comme outil d'exploration et de visualisation de données permet d'avoir un modèle décisionnel se rapprochant davantage de la réalité de l'utilisateur et lui demandant un moins grand effort d'abstraction, ce qui accroît son efficacité.

Dans la section suivante, nous allons intégrer cette composante spatiale dans l'entrepôt de données *N-Catch* afin de permettre une visualisation cartographique des résultats de simulation stockés, et de faciliter le processus d'aide à la décision.

3.5 Visualisation cartographique des données stockées dans *N-Catch*

La visualisation cartographique permet de mieux comprendre les phénomènes en voyant leur position dans un cadre de référence géographique, en voyant leur étendue dans l'espace et en voyant leur distribution dans l'espace (concentrée, dispersée, par groupes, aléatoire, régulière, etc.). Le couplage de composantes cartographiques et de

l'analyse en ligne nécessite de nouveaux outils clients. Deux technologies représentent des candidats potentiels : les systèmes d'information géographique (SIG) et le Spatial-OLAP ($S\text{-}OLAP$). Dans ce qui suit nous allons définir ces deux technologies et choisir celle qui correspond le mieux à notre cas d'étude.

3.5.1 Système d'information géographique (SIG)

Un système d'information géographique (SIG) [LGMR05] est un système d'information (SI) spécialisé dans le traitement de l'information géographique. L'information est géographique lorsqu'elle est liée à une localisation dans un système de référence sur la terre. On parle aussi de données localisées ou d'information à référence spatiale. Les SIG permettent d'assembler, de stocker, de manipuler et d'afficher l'information à référence spatiale. D'après Gardels [Gar97], un SIG doit répondre à 5 fonctionnalités "les 5 A" :

- **Abstraction** : L'abstraction représente la manière dont est modélisée l'information ;
- **Acquisition** : L'acquisition concerne la méthode de production (i.e. création et modification d'objets géographiques) des données ;
- **Archivage** : L'archivage représente le stockage et l'organisation des données géographiques ;
- **Analyse** : L'analyse est la partie représentant l'ensemble des outils d'analyse accessibles à l'utilisateur. Ces outils permettent, en plus de la simple visualisation de données géographiques, des analyses plus évoluées pour la prise de décision ;
- **Affichage** : L'affichage correspond à la restitution graphiques des données.

Dans la majorité des SIG, la partie Analyse est réduite ou inexistante. En effet, certains SIG peuvent être assimilés à des systèmes de cartographie numérique, tandis que, l'analyse est effectuée par des logiciels externes ou des *plugins*.

Nous nous intéressons en particulier dans cette thèse aux caractéristiques d'affichage des données géographiques.

3.5.2 Spatial-OLAP ($S\text{-}OLAP$)

Les entrepôts de données et l'analyse en ligne représentent une technologie clef et incontournable pour l'informatique décisionnelle. Les systèmes d'information géographiques, quant à eux, permettent de traiter des données géographiques, et de plus en plus, en grande quantité. Or, les outils d'analyse $OLAP$ ne disposent d'aucun outil pour l'analyse de données géographiques. À partir de ce constat, l'idée de coupler les deux technologies $OLAP$ et SIG a émergé. Ce couplage a donné naissance à une nouvelle famille d'outils mieux adaptée pour les analyses spatiales et spatio-temporelles, c'est-à-dire les technologies Spatial-OLAP.

Yvan Bédard présente la solution du Spatial-OLAP ($S\text{-}OLAP$ [BRP06]), comme : *"Une plateforme visuelle conçue spécialement pour supporter une analyse spatio-temporelle rapide et efficace à travers une approche multidimensionnelle qui comprend des niveaux d'agrégation cartographiques, graphiques et tabulaires"*.

Par conséquent, l'intérêt des systèmes *S-OLAP* est de pouvoir utiliser conjointement les outils *OLAP* (décision, graphiques, ...etc.) ainsi que les outils géographiques (représentation cartographique, agrégateurs géographiques,...etc.). Les systèmes *S-OLAP* permettent de gérer trois types de dimensions spatiales :

- Descriptives : les références spatiales sont textuelles (nom du lieu, par exemple) ;
- Géométriques : chaque niveau hiérarchique comprend un ensemble de formes géométriques (polygones, points, etc.) ;
- Mixtes : combinaison des deux, autorisant à la fois les références textuelles et géométriques.

Le traitement des dimensions géométriques et mixtes implique une redéfinition des opérateurs de navigation *OLAP* pour ce type de données. Plusieurs études ont proposé des opérateurs de navigation spatio-multidimensionnels (*Spatial Drill-down* et *Spatial-Roll-up*) [Bim07, BMH01]. Ces opérateurs ont le même fonctionnement que les opérateurs de forage basiques, mais sont appliqués à des objets géométriques.

Au début de la conception de *N-Catch*, nous ne disposions que de données textuelles (ex : numéro de la parcelle, type sol de la parcelle, numéro du bassin versant,...etc.) pour la description de la dimension spatiale. Ceci avait motivé le choix d'une implémentation *R-OLAP* (au lieu de *S-OLAP*) pour *N-Catch*. Par la suite, nous avons obtenu les données du cadastre du bassin versant du Yar afin de parvenir à visualiser les résultats des requêtes *OLAP* sur une carte. À ce stade, l'entrepôt de données *N-Catch* était déjà construit. Dans ce cas de figure, fallait-il ré-implémenter *N-Catch* avec *S-OLAP* afin d'exploiter ces nouvelles données spatiales ? Dans l'entrepôt de données *N-Catch*, la dimension spatiale ne contient que deux niveaux hiérarchiques (*Parcelle* et *Bassin-versant*). Finalement, nos besoins se résument à la visualisation cartographique des résultats de simulations. Nous pensons donc que l'utilisation d'opérateurs de navigation spatio-multidimensionnels n'est pas essentielle. Il est donc clair qu'un couplage *SIG-OLAP* était suffisant pour répondre à nos besoins. Grâce à ce couplage nous pouvons visualiser les résultats de simulation sur une carte sans avoir à modifier le contenu de l'entrepôt *N-Catch*. Dans ce qui suit, nous commençons par décrire les nouvelles données spatiales dont nous disposons (sous-section 3.5.3). Ensuite, nous présentons en sous-section 3.5.4 le processus de couplage de *N-Catch* avec le système d'information géographique *Quantum GIS* (*QGIS*), et des exemples de requêtes afin d'illustrer la visualisation cartographique via *QGIS*.

Ce travail a fait l'objet d'un stage professionnel de trois mois (du 1*er* octobre 2012 au 31 décembre 2012) de Sylvain DOUSSET que nous avons encadré.

3.5.3 Description des données spatiales

Les données spatiales correspondent au cadastre parcellaire du bassin versant du Yar. Elles ont été pré-traitées par l'UMR LETG, équipe COSTEL (Climat et Occupation du Sol par TELédétection) [2], qui a identifié 4620 parcelles sur le bassin du Yar (voir Figure.3.10). Ces données nous ont été fournies sous forme de fichiers au format vecteur

2. Composante de l'Université de Rennes 2 de l'UMR LETG - Littoral, Environnement, Télédétection, Géomatique

(shape) lisibles avec les systèmes d'information géographiques *ARCGIS* (payant) ou *QGIS* (gratuit).

FIGURE 3.10 – Carte du parcellaire du bassin versant du Yar

3.5.4 Couplage de *N-Catch* avec *QGIS*

Nous avons choisi de travailler avec le système d'information géographiques *QGIS* (voir site web : http ://www.qgis.org/). *QGIS* est un logiciel *SIG* publié sous licence *GPL*. Il fonctionne sous différents environnements informatiques et prend en charge de nombreux formats vectoriels, rasters ainsi que les formats et fonctionnalités de plusieurs bases de données. Un des principaux avantages de travailler avec *QGIS* est la possibilité d'y intégrer des extensions (*plugins*) pouvant enrichir ses fonctionnalités.

3.5.4.1 Jointure de *N-Catch* avec les données spatiales

Dans un *SIG*, chaque carte est associée à une table attributaire. Cette table contient des données alphanumériques qui décrivent les objets graphiques de la carte (ex : nom du propriétaire de la parcelle, type de sol de la parcelle,...etc.). Parmi les données contenues dans la table attributaire associée à la carte parcellaire du Yar, nous retrouvons l'identifiant de la parcelle. Cet identifiant véhicule la même information que l'attribut *Parcelle* de la dimension Localisation de l'entrepôt *N-Catch*. C'est donc cet identifiant qui va nous permettre de faire le lien (i.e. jointure) entre les données de la table attributaire et les données stockées dans *N-Catch*. Le but de cette jointure est de récupérer les résultats d'une requête posée à *N-Catch*, de réaliser une jointure avec la table attributaire, et à la fin d'afficher le résultat de cette jointure sur la carte parcellaire du Yar. Grâce à des scripts écrits en python (voir site web : http ://www.python.org/), on parvient à insérer les résultats des requêtes *OLAP* (i.e. les colonnes de la table résultats) dans la table attributaire et ce de manière dynamique. Par exemple, dans la Figure.3.11 nous visualisons sur la carte parcellaire les résultats de la requête *OLAP* qui calcule la moyenne des flux d'azote du sol vers la nappe à l'échelle de la parcelle sur toute la période de la simulation. Les parcelles les plus foncées représentent celles qui émettent le plus d'azote dans la nappe.

Pour faciliter l'interaction de l'usager avec *N-Catch* et l'outil de visualisation spatiale, nous avons réalisé un *plugin* avec une interface graphique permettant à l'utilisateur de sélectionner des données dans l'entrepôt *N-Catch* et de les visualiser sous forme cartographique. Ce *plugin* offre la possibilité à l'utilisateur soit de créer une nouvelle carte, ou plus précisément une nouvelle couche de données et de la sauvegarder (s'il le souhaite), soit de sélectionner une carte (couche de données) déjà existante (i.e. sauvegardée). Les couches de données sauvegardées représentent les vues matérialisées. Dans ce qui suit, nous présentons quelques exemples pour illustrer le fonctionnement de ce *plugin*.

3.5.4.2 Exemples de requêtes

Dans cet exemple, nous souhaitons visualiser la somme des flux quotidiens d'azote du sol vers la nappe pour l'année hydrologique 1997-1998 à l'échelle de la parcelle. Le résultat de cette requête est présentée dans la Figure.3.13 La première option qui s'offre à l'utilisateur est la création de nouvelles cartes, ou plus précisément de couches de données. Par exemple, dans la Figure.3.12, l'utilisateur a cliqué sur le bouton "Création

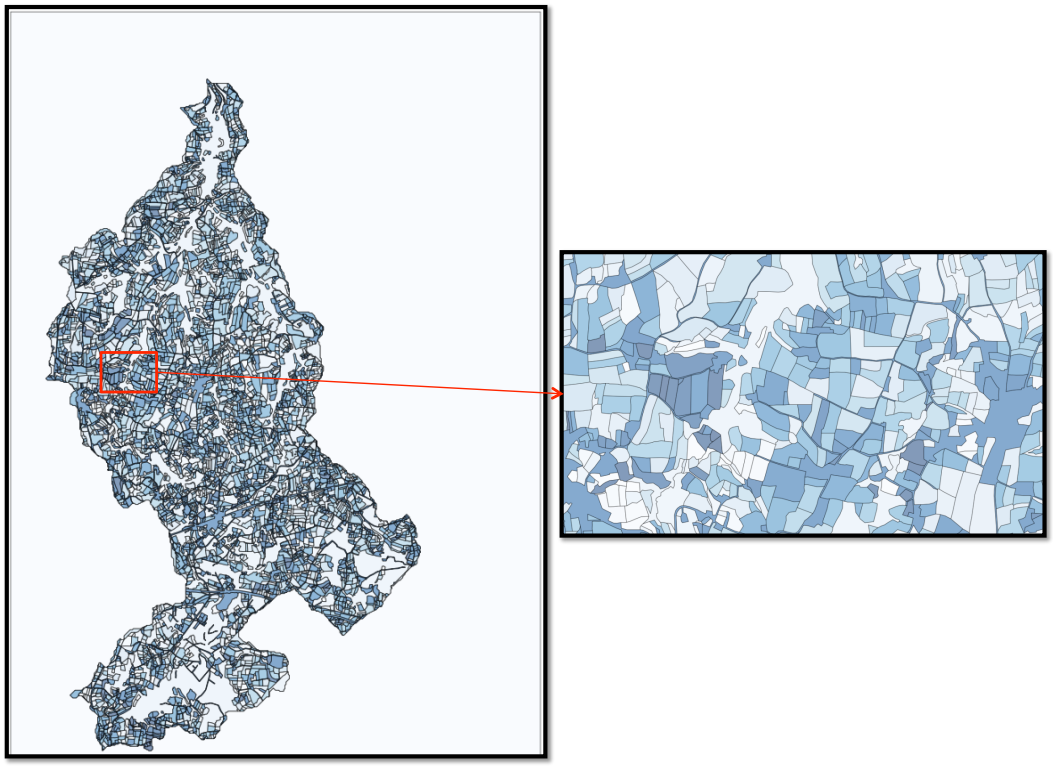


FIGURE 3.11 – Visualisation de la moyenne des flux d'azote du sol vers la nappe à l'échelle de la parcelle sur toute la période de la simulation sur la carte du parcellaire du Yar

Carte" du *plugin* ancré dans le bas du menu de gauche de l'interface de *QGIS*, et une fenêtre s'est ouverte. À partir de cette fenêtre, l'utilisateur peut sélectionner l'ensemble des données qu'il souhaite extraire de l'entrepôt de données. Les champs disponibles dans cette fenêtre représentent les niveaux hiérarchiques des dimensions de *N-Catch* les plus fréquemment interrogées (ex : le niveau *Culture* pour la dimension agricole, niveaux *Parcelle* pour la dimension spatiale,...). Cela dit, il est très facile de rajouter de nouveaux champs grâce aux scripts *python* existants.

La deuxième option qui s'offre à l'utilisateur est la sélection de couches de données déjà créées et sauvegardées. En effet, comme expliqué précédemment, nous avons matérialisé un ensemble de vues pour tenter de répondre de manière instantanée aux requêtes des utilisateurs. Par exemple, dans la Figure.3.14, l'utilisateur choisit d'afficher la carte parcellaire de la moyenne des flux d'azote du sol vers l'air pour la culture *Maïs* durant l'année 2003. Le résultat de cette requête est présentée dans la Figure.3.15.

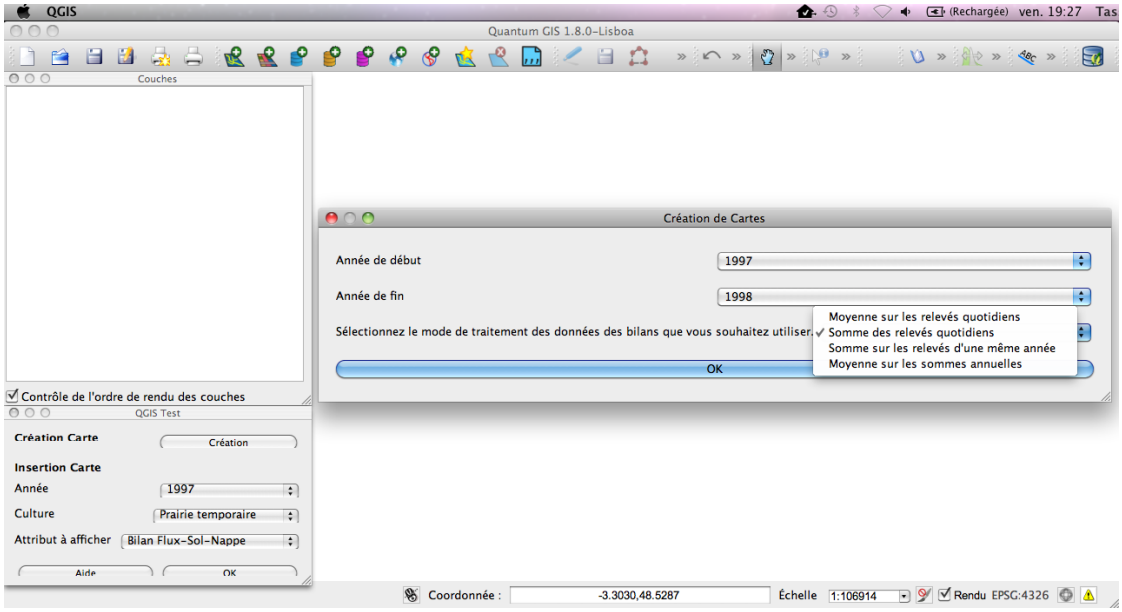


FIGURE 3.12 – Création d'une carte avec le nouveau *plugin*

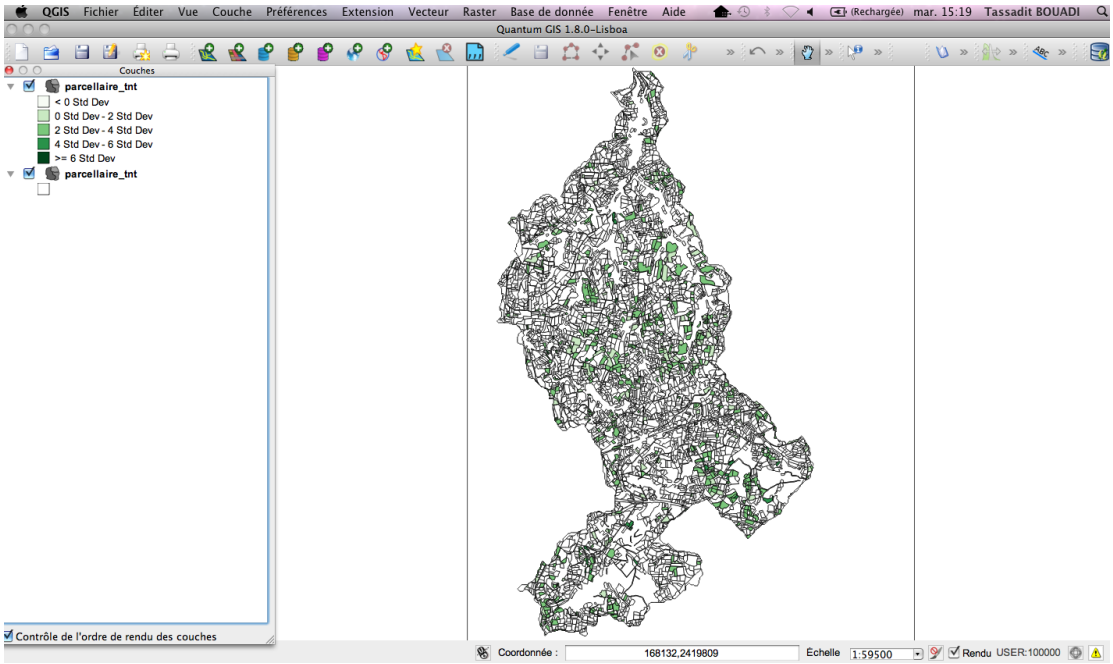


FIGURE 3.13 – Somme des flux quotidiens d'azote du sol vers la nappe à l'échelle de la parcelle pour l'année hydrologique 1997-1998 sur la carte du parcellaire du Yar

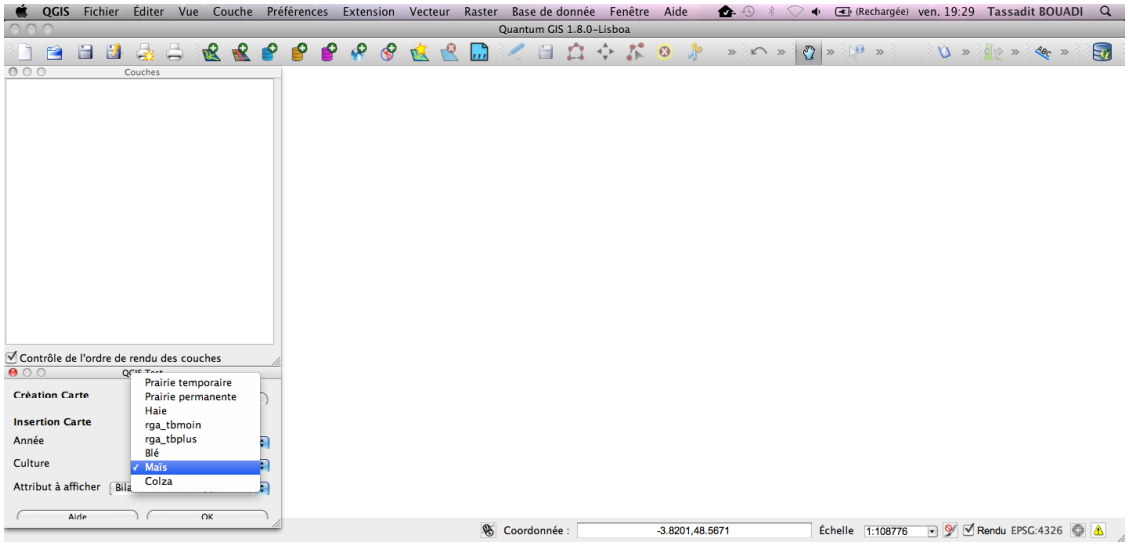


FIGURE 3.14 – Sélection d'une couche de données dans le *plugin*

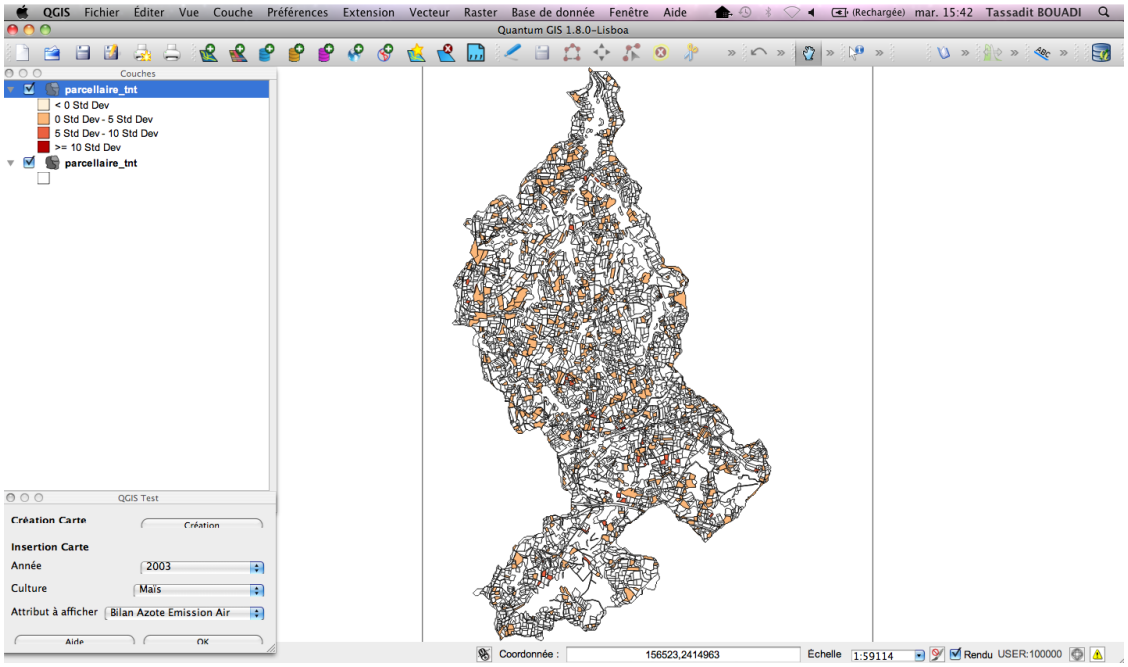


FIGURE 3.15 – Moyenne des flux d'azote du sol vers l'air à l'échelle de la parcelle pour la culture *Mais* durant l'année 2003 sur la carte du parcellaire du Yar

3.6 Conclusion

Dans ce chapitre, nous avons présenté le processus de construction de l'entrepôt de données *N-Catch*. *N-Catch* a été construit pour stocker les résultats de simulations issus du modèle *TNT*, et pour analyser le cycle de l'azote à différents niveaux spatio-temporels et de fournir des analyses stratégiques pour la prise de décision. Dans la table 3.2 nous présentons quelques chiffres relatifs au volume des données stockées dans *N-Catch*. L'un des défis majeur de cette étude résidait dans la quantité et la complexité des données à extraire, à traiter et à analyser. Comme mentionné précédemment, afin d'accélérer l'exécution des requêtes, nous avons utilisé : (i) des index qui nous ont permis de diviser par 5 les temps de calcul des requêtes portant sur les colonnes indexées, et (ii) des vues matérialisées qui assurent à des requêtes complexes et nécessitant seulement un accès aux vues matérialisées, avec donc des temps de réponses quasi instantanés.

Données stockées	Description
Entrées/sorites du modèle *TNT*	9000 fichiers traités (8.6 *Go*)
Taille de *N-Catch*	9 *Go* (sans les index et les vues matérialisées)
Index	2.1 *Go*
Vues matérialisées	2.7 *Go*

TABLE 3.2 – Synthèse des données stockées dans *N-Catch*

Dans ce travail nous avons proposé une méthodologie de construction d'un entrepôt de données agro-hydrologique, dont les principales contributions sont :

- Le pré-traitement et la transformation des données extraites ;
- La modélisation multidimensionnelle et hiérarchique des pratiques agricoles ;
- l'analyse des résultats de simulations en combinant la modélisation spatio-temporelle des données, l'entreposage des données et l'analyse en ligne.

Ces contributions méthodologiques peuvent être appliquées à une variété de problématiques agro-environnementales. Plus généralement, cette approche peut être utilisée pour analyser l'impact des pratiques agricoles sur la qualité des eaux, ou d'autres impacts environnementaux. Les différentes phases de conception et de construction de l'entrepôt de données (identification des besoins, modélisation multidimensionnelle, et utilisation de OLAP pour accéder et exploiter des données multidimensionnelles et agrégées) sont des étapes génériques.

L'entrepôt de données *N-Catch* peut être utilisé efficacement pour :

- Explorer les dimensions spatio-temporelles et exploiter les simulations afin de fournir une analyse plus fine pour faciliter le processus d'aide à la décision (i.e. rendre les données facilement accessibles par les décideurs) ;
- Permettre aux utilisateurs de synthétiser l'information environnementale et de comprendre les émissions d'azote dans les cours d'eau,
- Analyser les processus agro-hydrologiques et extraire les relations entre les mesures.

Cependant, avec l'analyse $OLAP$, l'exploration est faite par l'utilisateur mais sans outil pour le guider automatiquement dans l'entrepôt de données $N\text{-}Catch$. C'est à l'utilisateur de décider vers quelles données naviguer à la recherche d'informations intéressantes et c'est aussi à l'utilisateur d'évaluer la pertinence des informations découvertes pour savoir si elles constituent de nouvelles connaissances. Pourtant, l'automatisation de ces tâches serait très utile à l'utilisateur de $N\text{-}Catch$.

Dans la partie suivante, nous proposons de coupler $N\text{-}Catch$ avec les requêtes Skyline afin de permettre aux utilisateurs de formuler de nouvelles requêtes dans $N\text{-}Catch$ en combinant des indicateurs environnementaux contradictoires, et de trouver les solutions compromis associées à ces attentes. Nous allons montrer comment exploiter les préférences des utilisateurs afin de détecter les données susceptibles de les intéresser.

Troisième partie

Requêtes skyline dans un contexte multidimensionnel et hiérarchique

Chapitre 4

Calcul incrémental des requêtes skyline en présence de préférences dynamiques

4.1 Introduction et motivations

Un nombre considérable de méthodes de calcul de skyline a été proposé dans le cadre des bases de données. La quasi totalité de ces méthodes étaient associées à des dimensions totalement ordonnées. Cependant, dans les applications réelles, les données peuvent inclure des dimensions partiellement ordonnées par nature, comme les dimensions catégoriques (i.e. nominales). Par exemple, dans notre application agro-hydrologique (cf. chapitre 3), le niveau type culture de la dimension agricole est catégorique. Certaines études récentes se sont intéressées à la problématique d'extraction de points skyline en présence de dimensions partiellement ordonnées [BGS06], et ont fourni un moyen pour vérifier la dominance entre deux points incomparables pour un ordre partiel donné. Cependant, ces travaux supposent qu'il existe un seul ordre prédéfini sur le domaine de chaque dimension. En particulier, les utilisateurs ne peuvent pas exprimer en ligne des préférences entre les différentes dimensions ni personnaliser les préférences entre les valeurs d'une dimension donnée, et calculer les points skyline associés à ces préférences. Par conséquent, d'autres types de requêtes skyline ont été proposées pour traiter ce problème :

– les préférences inter-dimensions qui permettent à l'utilisateur de spécifier le degré d'importance de différentes dimensions et ainsi de classer les skyline par ordre de préférences [MC11]. Par exemple, la dimension agricole peut être considérée comme plus importante que la dimension spatiale si notre objectif est de caractériser les types de cultures qui polluent le plus. Ce type de requêtes permet aussi de réduire le nombre de points skyline qui grandit de manière exponentielle avec le nombre de dimensions, en retournant seulement les K meilleurs points, les top K [BGG07], associés aux préférences inter-dimensions définies.

– les préférences intra-dimensions [WFP$^+$08, WPFW09] qui permettent à chaque utilisateur d'exprimer des préférences sur les différentes valeurs d'une dimension. Ce type de préférences est particulièrement intéressant pour les dimensions nominales où il n'est pas évident de trouver un ordre consensuel. Par exemple, un utilisateur peut préférer extraire les parcelles les plus polluées avec comme type de culture *prairie temporaire*, et un autre utilisateur peut plutôt préférer extraire les parcelles avec le type de culture *CIPAN*.

Dans cette étude, nous nous intéressons particulièrement au calcul en ligne des requêtes skyline en présence de préférences intra-dimensions. Un problème intéressant se pose lorsque les utilisateurs ont la possibilité de définir ou de modifier leurs préférences en ligne. Ainsi, sur certaines dimensions l'ordre peut changer dynamiquement. Ce problème constitue un véritable challenge dans le contexte des bases de données volumineuses comme les entrepôts de données, et il a attiré l'attention de travaux récents [WFP$^+$08, WPFW09]. En effet, le skyline évolue lorsque les préférences changent. Une solution naïve serait de recalculer, pour chaque nouvelle préférence, l'ensemble des skyline. Cependant, ceci devient trop coûteux dans un contexte de données volumineuses avec une dimensionalité élevée. Le défi est donc le suivant : comment recalculer efficacement le moins de points skyline possible, tout en minimisant l'espace mémoire requis.

Les solutions proposées par Wong et al. dans [WPFW09, WFP$^+$08] développent des méthodes de semi-matérialisation (i.e. matérialisation partielle des préférences et des skyline associés) assurant un calcul en ligne des requêtes skyline impliquant des préférences dynamiques. Précisément, les auteurs de [WFP$^+$08] ont introduit le concept de *préférence d'ordre n*. Ils ont démontré que le skyline associé à n'importe quelle préférence sur une dimension donnée pouvait être calculé à partir des préférences de premier ordre sur cette même dimension. S'appuyant sur cette propriété (nommée *merging property*), ils proposent de matérialiser les skyline associés aux préférences de premier ordre dans une structure de données spécifique, nommée *IPO-tree*, afin d'accélérer le calcul en ligne des skyline. Pour gérer le cas de multiples dimensions, ils proposent de stocker toutes les combinaisons possibles des préférences de premier ordre dans la structure *IPO-tree*. Par conséquent, la taille de *IPO-tree* induite par cette matérialisation est de l'ordre de $O(c^m)$, où m représente le nombre de dimensions associées à des préférences dynamiques, et c la cardinalité maximale d'une dimension. Dans le contexte de bases de données volumineuses et multidimensionnelles, la taille et la gestion de cette structure de données est prohibitive en terme d'espace mémoire et de temps de calcul.

Dans [WPFW09], Wong et al. ont proposé la structure CST (*Compressed Ordered skyline Tree*), pour matérialiser tous les ordres de préférence possibles. Ils stockent les skyline associés aux divers ordres de raffinement dans une structure de données compacte. Cependant, l'arbre CST construit est très complexe. Cela rend difficile les mises à jour des préférences qui nécessitent une lourde maintenance et des modifications conséquentes au niveau de l'arbre CST. De plus, cette méthode est incomplète. En effet, l'arbre CST est construit graduellement en ajoutant, une par une, les dimensions dynamiques. Au cours de cette construction, certains points sont disqualifiés du skyline

lors de l'ajout d'une nouvelle dimension, alors qu'ils devraient être dans le skyline. Nous avons notifié ce problème d'incomplétude aux auteurs, et un erratum sera publié prochainement dans [WPFW] (communication personnelle de Wong).

La *merging property* de Wong et al. ne gère qu'une seule dimension à la fois, et par conséquent, est d'un intérêt limité.

Dans ce chapitre, nous étudions le cas d'un nombre arbitraire de dimensions. Notre proposition, EC^2Sky [BCQ12, BCQ13], se focalise sur comment répondre efficacement à des requêtes de type skyline en présence de préférences utilisateurs dynamiques sur plusieurs dimensions malgré de gros volumes de données.

L'idée principale d'EC^2Sky repose sur l'ajout incrémental des dimensions dynamiques lors du calcul des skyline. Comme effet supplémentaire, EC^2Sky retourne les connaissances les plus pertinentes en soulignant les compromis associés aux préférences spécifiées. L'avantage de cette proposition est double. D'une part, la complexité en terme de stockage mémoire des informations pré-calculées est réduite à $O(c * m)$ (où m représente le nombre de dimensions associées à des préférences dynamiques, et c la cardinalité maximale d'une dimension). D'autre part, le nombre de tests de dominance diminue de manière significative. De l'espace mémoire et des temps d'exécution supplémentaires sont nécessaires au calcul des skyline associés aux préférences de premier ordre comparé à la méthode *IPO-tree*. Mais nous avons expérimentalement prouvé que le coût total de calcul est beaucoup plus faible que pour *IPO-tree*. Cette contribution permet un calcul incrémental des skyline associés à un ensemble de préférences, et facilite la modification interactive de ces dernières.

Dans ce chapitre, nous présentons dans la section 4.2 les concepts de base liés aux préférences dynamiques et les principales méthodes de calcul de skyline en présence de telles préférences. Dans la section 4.3, nous développons les aspects formels ainsi que l'implémentation de notre approche EC^2Sky, et nous présentons les résultats de l'évaluation expérimentale réalisée sur des données synthétiques qui souligne la pertinence de la solution proposée en la comparant aux références du domaine.

4.2 Skyline et préférences dynamiques

Les différentes définitions fournies dans cette section sont illustrées à partir de l'exemple de la table 4.1, décrivant un ensemble de parcelles d'un bassin versant agricole selon, le type de sol (Ts), la durée moyenne d'une prairie dans une rotation culturale (Dp), le rendement (Re) (kg/ha) et le cumul d'émissions d'azote du sol vers la nappe (Sn)($kgN/ha/année$).

Définition 6 ***(Type de préférences)*** *Nous distinguons deux types de préférences :*
- *préférences statiques : elles correspondent à une relation d'ordre prédéfinie,*
- *préférences dynamiques : elles correspondent à une relation d'ordre pouvant varier d'un utilisateur à un autre ou encore d'une session d'un même utilisateur à une autre.*

ID parcelle	Ts	Dp	Sn	Re
a	SM(Sol sain micaschiste)	PP(Prairie-permanente)	16	200
b	SM(Sol sain micaschiste)	PP(Prairie-permanente)	24	500
c	H(Zone humide)	PP(Prairie-permanente)	30	100
d	H(Zone humide)	PC(Prairie-courte)	36	200
e	SM(Sol sain micaschiste)	PC(Prairie-courte)	23	400
f	SG(Sol sain granite)	PM(Prairie-moyenne)	30	300
g	SG(Sol sain granite)	PC(Prairie-courte)	36	200
h	SG(Sol sain granite)	PC(Prairie-courte)	30	300

TABLE 4.1 – Valeurs des dimensions des parcelles d'un bassin versant agricole

Par abus de langage, nous utilisons l'expression *dimensions dynamiques (resp. statiques)* à la place de l'expression *dimensions associées avec des préférences dynamiques (resp. statiques).*

Dans la suite de ce chapitre, on note par S l'ensemble des dimensions statiques de l'espace D, et par Z l'ensemble des dimensions dynamiques de l'espace D, avec $D = S \bigcup Z$ et $S \bigcap Z = \varnothing$.

Exemple 9 *$S = \{Sn, Re\}$: les valeurs de ces deux dimensions suivent la relation d'ordre $\leqslant$ indiquant que plus le taux de pollution est bas (resp. plus le rendement est élevé), plus la parcelle est préférable (resp. moins la parcelle est préférable) (ex : $a(Sn) <_{Sn} d(Sn)$). Cet ordre est valable pour tous les utilisateurs, il est donc statique. Pour $Z = \{Ts, Dp\}$, aucune relation d'ordre n'est définie a priori (i.e., de manière statique) sur ces dimensions. La définition d'un ordre est laissée à l'utilisateur et peut varier d'un utilisateur à un autre.*

Soit un utilisateur souhaitant identifier les parcelles ayant les meilleurs rendements et émettant le moins d'azote dans la nappe. Dans ce cas :
$Sky(S, E) = \{a, b, e\}$.
$Sky(D, E) = \{a, b, e, c, d, f, h\}$ lorsqu'aucune préférence n'est spécifiée sur les dimensions Ts et Dp. Les points c, d, f et h ne sont plus dominés par les points skyline a, b et e, étant donné qu'aucun point ne peut dominer sur les dimensions non ordonnées Ts et Dp.

Les définitions qui suivent sont posées sur un sous ensemble $D' \subseteq D$ et sont donc généralisables à l'ensemble D.

Dans cette étude, nous souhaitons aider l'utilisateur dans son exploration de l'ensemble des données en lui permettant d'exprimer diverses préférences sur les dimensions dynamiques et d'évaluer les conséquences de ces choix en retournant les meilleurs points i.e. les points skyline. Par exemple, soit type de sol (Ts) une dimension dynamique. Différents utilisateurs peuvent exprimer différentes préférences sur cette dimension. Si un utilisateur donne la préférence aux parcelles en zones humides (i.e. avec le type

de sol H) plutôt que les autres. Les parcelles c, d, f et h sont alors rajoutées dans $Sky(\{Sn, Re, Ts\}, E)$ puisqu'elles ont les meilleures valeurs sur la dimension Ts. Cependant, un autre utilisateur peut donner la préférence aux parcelles cultivées sur sols sains en micaschiste (i.e. avec le type de sol SM) plutôt que les autres. Dans ce cas, les parcelles c, d, f et h ne font pas partie du skyline car elles sont dominées par a, b et e. Il est intéressant d'observer que les parcelles associées aux identifiants a, b et e appartiennent toujours au skyline quelle que soit la préférence choisie sur la dimension Ts (car a est le seul à avoir la meilleure valeur sur la dimension Sn, b est le seul à avoir la meilleure valeur sur la dimension Re et e constitue un bon compromis pour les deux dimensions Sn et Re).
Lorsqu'un utilisateur formule une requête impliquant une dimension dynamique d_i, elle/il peut spécifier l'ordre de préférence sur les valeurs de cette dimension. L'ordre est total si toutes les valeurs sont ordonnées. Cependant, cela n'est pas toujours possible et l'utilisateur peut ordonner seulement n valeurs sur les $|d_i|$ valeurs. Implicitement, elle/il considère que ces n valeurs sont préférées aux $(|d_i| - n)$ valeurs restantes qui sont laissées non ordonnées. Ceci correspond à la notion de *préférence implicite d'ordre n* introduite dans [WFP+08].

Définition 7 ***(Préférence d'ordre** n**)*** *Soit $d_i \in Z$ et $|d_i| = m$. $\wp_i$ est une préférence d'ordre n sur d_i ssi :*

- $\wp_i = v_1 <_{d_i} \ldots <_{d_i} v_n <_{d_i} *$, *avec* $v_1 \in dom(d_i), \ldots, v_n \in dom(d_i)$ *et* $n \leqslant m$,
- $\forall k \in \{n+1, .., m\}, v_n <_{d_i} v_k$.

Lorsque $n = 1$, $\wp_i = v_1 <_{d_i} *$ est appelée préférence de premier ordre.

Ainsi, $<_{d_i}$ représente un ordre total sur les valeurs $\{v_1 \ldots v_n\}$ de d_i, et un ordre partiel sur l'ensemble du domaine de valeurs de d_i.
À noter l'importance des préférences de premier ordre : elles sont suffisantes pour déterminer les points dominants sur une dimension.

Exemple 10 *Pour la dimension Type de sol de la table 4.1, un utilisateur préfère SM à SG, SM à H et SG à H (i.e. $SM <_{Ts} SG <_{Ts} H$). Cette préférence représente une préférence d'ordre* 3 *et définit un ordre total. Un utilisateur peut préférer le type de sol SM à tous les autres types (i.e. $SM <_{Ts} *$). Ceci correspond à une préférence de premier ordre qui définit l'ordre partiel $\{SM <_{Ts} SG, SM <_{Ts} H\}$.*

$\wp_i = v_1 <_{d_i} \ldots <_{d_i} v_n <_{d_i} *$ désigne l'ensemble des préférences binaires $\wp_i = \{v_1 <_{d_i} v_2, v_2 <_{d_i} v_3, \ldots, v_n <_{d_i} *\}$. Dans la suite, nous utilisons les deux notations pour $\wp_i$.

Exemple 11 *Soit $Z = \{Ts\}$, $\wp = \{H <_{Ts} *\}$ (équivalent à $\{H <_{Ts} SM, H <_{Ts} SG\}$). Alors, $Sky(D, E)_{(Z, H <_{Ts} *)} = \{a, b, e, c, d, f, h\}$. Les points c et d ne sont pas dominés par les points skyline $\{a, b, e\}$ (cf. exemple 9) puisqu'ils ont les meilleures valeurs sur la dimension Ts. Les valeurs SM et SG de la dimension Ts sont laissées non ordonnées, ce qui permet aux points f et h de devenir des points skyline compromis étant donné qu'ils ne sont pas dominés par les points skyline $\{a, b, e\}$ sur la dimension Ts.*

Nous présentons ci-dessous quelques propriétés utiles de la relation de préférence. Ces propriétés seront utilisées par la suite pour réduire le nombre de tests de dominance lors du calcul des skyline. Dans la suite, $\wp = \bigcup_{i=1}^{|Z|} \wp_i$ représente l'ensemble des préférences associées aux dimensions dynamiques de $Z \subseteq D$ et l'ensemble des préférences associées aux des dimensions statiques de $S \subseteq D$ (préférences présentes implicitement).

Définition 8 ***(Raffinement d'une préférence)*** *Soient $\wp'$ et $\wp''$ deux ensembles de préférences sur le sous-espace Z. $\wp''$ est un raffinement de $\wp'$ si $\wp' \subseteq \wp''$.*

Propriété 1 ***(Monotonie du raffinement d'une préférence)*** *Soient $\wp'$ et $\wp''$ deux ensembles de préférences sur le sous-espace Z. Si $\wp''$ est un raffinement de $\wp'$, alors $Sky(D,E)_{(Z,\wp'')} \subseteq Sky(D,E)_{(Z,\wp')}$.*

L'exemple suivant illustre la propriété 1.

Exemple 12 *Soient $Z = \{Ts\}$, $\wp' = \{H <_{Ts} *\}$ et $\wp'' = \{H <_{Ts} SM <_{Ts} *\}$. $\wp''$ est un raffinement de $\wp'$ puisque $\wp' \subset \wp''$.*
$Sky(D,E)_{(Z,\wp')} = \{a,b,c,d,e,f,h\}$ et $Sky(D,E)_{(Z,\wp'')} = \{a,b,c,d,e\}$.
Nous avons bien : $Sky(D,E)_{(Z,\wp'')} \subset Sky(D,E)_{(Z,\wp')}$.

La propriété 1 indique que lorsque les préférences sont raffinées, le skyline peut devenir plus petit (i.e. certains points skyline peuvent être disqualifiés). De plus, si un point n'appartient pas au skyline associé à une préférence donnée, il n'appartiendra pas non plus au skyline associé à un raffinement de cette préférence. Nous utiliserons, dans notre approche, la propriété 1 pour réduire le nombre de tests de dominance.

Le théorème suivant formule une propriété importante appelée *merging property*. Cette dernière a été introduite par Wong et al. dans [WFP+08]. Cette propriété permet de dériver le skyline de toutes les préférences d'ordre n possibles sur une dimension en réalisant des opérations sur les préférences de premier ordre sur cette même dimension.

Théorème 1 ***(Merging property)*** *Soient $\wp^{'}$ et $\wp^{''}$ deux ensembles de préférences qui diffèrent seulement sur la dimension d_i, i.e. $\wp^{'}_j = \wp^{''}_j$ pour tout $j \neq i$. De plus, $\wp^{'}_i = v_1 <_{d_i} \ldots < v_{k-1} <_{d_i} *$ et $\wp^{''}_i = v_k <_{d_i} *$. Soit $PSky(D,E)_{(Z,\wp')}$ l'ensemble des points dans $Sky(D,E)_{(Z,\wp')}$ avec des valeurs de d_i dans $\{v_1 \ldots v_{k-1}\}$. Soit $\wp^{'''}$ un ensemble de préférences qui ne diffère de $\wp^{'}$ et $\wp^{''}$ que sur la dimension d_i et $\wp^{'''}_i = v_1 < \ldots < v_{k-1} < v_k < *$. Le skyline associé à $\wp^{'''}$ est alors :*

$$Sky(D,E)_{(Z,\wp''')} = (Sky(D,E)_{(Z,\wp')} \bigcap Sky(D,E)_{(Z,\wp'')}) \bigcup PSky(D,E)_{(Z,\wp')}.$$

Exemple 13 *Soient $\wp^{'} = \{SG <_{Ts} *\}$, $\wp^{''} = \{H <_{Ts} *\}$, $\wp^{'''} = \{SG <_{Ts} H <_{Ts} *\}$ et $Z = \{Ts\}$.*
$Sky(D,E)_{(Z,\wp''')} = (Sky(D,E)_{(Z,\wp')} \bigcap Sky(D,E)_{(Z,\wp'')}) \bigcup PSky(D,E)_{(Z,\wp')}$
$= (\{a,b,e,f,h\} \bigcap \{a,b,e,c,d,f,h\}) \bigcup \{f,h\} = \{a,b,e,f,h\}$

Wong et al. ont proposé une méthode intéressante, *IPO-tree* [WFP+08], pour le calcul des skyline en présence de préférences dynamiques en s'appuyant sur la *merging property*. Dans ce qui suit, nous présentons cette méthode de manière plus détaillée, et discutons ces points forts et ces points faibles.

4.2.1 Algorithme $IPO\text{-}Tree$

Wong et al. [WFP+08] ont proposé une méthode de semi-matérialisation basée sur une structure de données spécifique nommée $IPO\text{-}tree$ (*Implicit Preference Order Tree*). $IPO\text{-}Tree$ stocke seulement les résultats partiels correspondant à toutes les combinaisons possibles des préférences de premier ordre des différentes dimensions dynamiques. Wong et al. ont également introduit la propriété 1 qui permet de dériver le skyline de n'importe quelle préférence d'ordre n à partir des résultats partiels stockés. Cependant, cette approche présente un inconvénient majeur. En effet, la taille de la structure $IPO\text{-}Tree$ est de l'ordre de $O(c^m)$, où m représente le nombre de dimensions associées à des préférences dynamiques, et c la cardinalité maximale d'une dimension. Cette méthode ne résiste donc pas au passage à l'échelle, et devient vite inexploitable dans un contexte de données volumineuses. De plus, il est intéressant de noter que la propriété 1 s'applique à une seule dimension à la fois.

Tout en réutilisant la propriété 1 proposée par Wong et al. permettant de traiter les raffinements des préférences sur une seule dimension, nous proposons une méthode incrémentale, appelée EC^2Sky, pour le calcul des skyline associés à plusieurs dimensions dynamiques. En effet, le skyline peut changer lorsque les préférences sont mises à jour, et devrait donc être progressivement maintenu pour éviter une réévaluation globale du skyline. Contrairement à la méthode $IPO\text{-}tree$, EC^2Sky fournit à l'utilisateur un moyen d'exprimer des préférences et de les modifier en ligne sans être pénalisé par des temps d'attente trop longs. Les performances sont obtenues en ne stockant que le minimum d'informations requises afin d'autoriser des mises à jour simples et rapides.

Dans ce qui suit, nous détaillons plus précisément notre proposition d'une méthode efficace de calcul de skyline en présence de préférences dynamiques.

4.3 Algorithme EC^2Sky

Dans cette section, nous introduisons la méthode incrémentale proposée ainsi que le théorème sur lequel la méthode est fondée. Dans la suite de ce chapitre, nous supposons que le sous-espace de dimensions D^i est tel que $D^i = D^{i-1} \bigcup d_i$, avec $d_i \in Z$, $i \in \{1, .., |Z|\}$, $D^i \subseteq D$ et $D^0 = S$. Cette notation représente l'ajout incrémental de dimensions lors du calcul des skyline.

Maintenant, examinons comment l'ajout d'une dimension dynamique d_i influe sur le skyline qui a déjà été calculé pour le sous-espace de dimensions D^{i-1}. Intuitivement, le calcul du nouveau skyline du sous-espace $D^i = D^{i-1} \bigcup d_i$ se déroule en deux étapes. La première étape, consiste à calculer le skyline associé à la nouvelle dimension dynamique et à l'ensemble des dimensions statiques $d_i \bigcup S$, comme si elle était indépendante des

autres dimensions dynamiques. La seconde étape, prend en compte les conséquences de l'introduction de la nouvelle dimension d_i à l'ensemble des dimensions précédentes D^{i-1} pour mettre à jour le nouveau skyline. Cette seconde tâche consiste à, (i) retirer du skyline, indépendamment calculé pour $d_i \bigcup S$, les points qui sont disqualifiés i.e. sont dominés sur les dimensions dynamiques de D^{i-1}, (ii) retirer l'ensemble des points skyline de D^{i-1} qui sont disqualifiés i.e. sont dominés sur la nouvelle dimension d_i, et (iii) compléter le skyline résultant avec les nouveaux points skyline compromis sur $D^{i-1} \bigcup d_i$.

Dans ce qui suit, nous décrivons ces trois étapes de manière plus détaillée.

4.3.1 Calcul incrémental des skyline : Théorème

Considérons l'ajout de la dimension dynamique d_i à l'ensemble D^{i-1} de $i-1$ dimensions dynamiques. Comme expliqué ci-dessus, la première étape consiste à calculer $Sky(d_i \bigcup S, E)_{(Z,\wp)}$. La méthode de Wong et al. peut être utilisée pour réaliser cette tâche. Cependant, cet ensemble peut contenir des points qui sont normalement disqualifiés i.e. qui sont dominés sur les dimensions dynamiques du sous-espace D^{i-1}. Précisément, soient $p, q \in Sky(d_i \bigcup S, E)_{(Z,\wp)}$ deux points skyline ayant les mêmes valeurs sur chaque dimension de $d_i \bigcup S$. Si q est préféré sur D^{i-1}, il dominera alors p et le disqualifiera du skyline $Sky(D^i, E)_{(Z,\wp)}$. Cet ensemble de points est nommé $CutSky(d_i \bigcup S, E)$.

Définition 9 ***(Les points skyline disqualifiés de*** $d_i \bigcup S$***)*** *L'ensemble des points skyline associé au sous-espace $d_i \bigcup S$ qui est disqualifié par l'introduction du sous-espace D^{i-1} est défini par :*
$CutSky(d_i \bigcup S, E) =$
$\{p \in Sky(d_i \bigcup S, E)_{(Z,\wp)} \mid \exists q \in Sky(d_i \bigcup S, E)_{(Z,\wp)},\ p =_{d_i \bigcup S} q \wedge q \prec_{D^{i-1}} p\}$.

Exemple 14 *Soient $D^{i-1} = D^1 = \{Sn, Re, Dp\}$, $d_i = Ts$, la nouvelle dimension, et les préférences : $\{SG <_{Ts} H <_{Ts} SM\}$ et $\{PM <_{Dp} *\}$.*
$Sky(D^1, E)_{(Z,\wp)} = \{a, b, e, f\}$ *et* $Sky(\{Ts\} \bigcup S, E)_{(Z,\wp)} = \{a, b, e, f, h\}$.
$CutSky(\{Ts\} \bigcup S, E) = \{h\}$ *car le point h doit être retiré du skyline $Sky(D^2, E)_{(Z,\wp)}$, étant donné qu'il est dominé par le point f sur le sous-espace D^1.*

D'autre part, l'ancien skyline $Sky(D^{i-1}, E)_{(Z,\wp)}$ peut contenir des points qui sont disqualifiés par des points dominants introduits par la nouvelle dimension d_i. Précisément, soient $p, q \in Sky(D^{i-1}, E)_{(Z,\wp)}$ deux points skyline ayant les mêmes valeurs sur chaque dimension de D^{i-1}. Si q est préféré sur la nouvelle dimension d_i, il dominera alors p et le disqualifiera du skyline $Sky(D^i, E)_{(Z,\wp)}$. Cet ensemble de points est nommé $CutSky(D^{i-1}, E)$.

Définition 10 ***(Les points skyline disqualifiés de*** D^{i-1}***)*** *L'ensemble des points skyline associé au sous-espace D^{i-1} qui est disqualifié par l'introduction de la nouvelle dimension d_i est défini par :*
$CutSky(D^{i-1}, E) =$
$\{p \in Sky(D^{i-1}, E)_{(Z,\wp)} \mid \exists q \in Sky(D^{i-1}, E)_{(Z,\wp)}, p =_{D^{i-1}} q \wedge q \prec_{d_i \bigcup S} p\}$.

Exemple 15 *Soient $D^{i-1} = D^1 = \{Sn, Re, Ts\}$, $d_i = Dp$, la nouvelle dimension, et les préférences $\{SG <_{Gr} H <_{Gr} SM\}$ et $\{PP <_{Dp} PC <_{Dp} PM\}$.*
$Sky(D^1, E)_{(Z,\wp)} = \{a, b, e, f, h\}$, $CutSky(\{Dp\} \bigcup S, E) = \{\}$ et
$CutSky(D^1, E) = \{f\}$ car le point f doit être retiré du skyline $Sky(D^2, E)_{(Z,\wp)}$, étant donné qu'il est dominé par le point h sur la nouvelle dimension $d_i = Dp$.

Finalement, certains points devraient apparaître dans le nouveau skyline. Précisément, avant de prendre en compte la nouvelle dimension d_i, certains points peuvent être dominés sur chaque dimension de $D^{i-1} \bigcup S$, et donc n'appartiennent pas au skyline. Mais, lorsque la dimension d_i est introduite, ils deviennent meilleurs sur la dimension d_i que certains points skyline qui les dominaient avant. Ces points ne sont plus dominés par aucun autre point skyline de $Sky(D^{i-1}, E)_{(Z,\wp)}$ sur certaines dimensions de D^{i-1} : ils constituent les nouveaux points skyline compromis. Cet ensemble de points est nommé $NewCompSky(D^i, E)$.

Définition 11 ***(Nouveaux skyline compromis)***
Soit $C = Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)}$.
L'ensemble des nouveaux skyline compromis est défini par
$NewCompSky(D^i, E) = \{p \in E - C \mid \forall q \in E, \exists d_k \in D^i, p \prec_{d_k} q\}$.

Exemple 16 *Soient $D^{i-1} = D^1 = \{Sn, Re, Ts\}$, $d_i = Dp$, la nouvelle dimension, et les préférences $\{SG <_{Gr} H <_{Gr} SM\}$ et $\{PP <_{Dp} PC <_{Dp} PM\}$.*
$Sky(D^1, E)_{(Z,\wp)} = \{a, b, e, f, h\}$, $Sky(\{Dp\} \bigcup S, E)_{(Z,\wp)} = \{a, b, e\}$,
$CutSky(\{Dp\} \bigcup S, E) = \{\}$ et $CutSky(D^1, E) = \{f\}$.
Mais, si on considère simultanément les deux dimensions Ts et Dp, alors c n'est plus dominé par f. Étant donné que f était le seul qui dominait c, c devient un nouveau point skyline. Comme c est le seul point "promu", $NewCompSky(D^2, E) = \{c\}$.

Supposons que nous souhaitons étendre le sous-espace de dimensions D^{i-1} avec une nouvelle dimension d_i. Le théorème suivant indique que le skyline du sous-espace étendu peut être calculé en retirant les points skyline disqualifiés de l'ancien skyline, et en ajoutant les nouveaux points skyline introduits par la préférence associée à la nouvelle dimension. Les nouveaux points skyline sont soit des points dominants sur la nouvelle dimension soit des nouveaux skyline compromis introduits par la nouvelle préférence.

Théorème 2 ***(Skyline incrémental)***
Soient E *un ensemble de points à $|D|$ dimensions, $Z \subseteq D$ le sous-espace de taille $|Z| = m$ avec des préférences dynamiques $\wp = \{\wp_j\}_{j=1,..,m}$ sur D, $Sky(d_i \bigcup S, E)$ le skyline du sous-espace $S \bigcup d_i$ et $D^i = D^{i-1} \bigcup d_i$, avec $i = \{1, .., m\}$.*

$Sky(D^i, E)_{(Z,\wp)} = (Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)}) - (CutSky(D^{i-1})$
$\bigcup CutSky(d_i \bigcup S)) \bigcup NewCompSky(D^i, E)$.

Exemple 17 ***(Illustration du théorème 2)*** *Soient* $D^1 = \{Sn, Re, Ts\}$, $D^2 = \{Sn, Re, Ts, Dp\}$ *et les préférences de l'exemple précédent.*
$Sky(D^2, E)_{(Z,\wp)} =$
$(Sky(D^1, E)_{(Ts, SG<_{Ts}H<_{Ts}SM} \bigcup Sky(\{Dp\} \bigcup S, E)_{(Dp, PP<_{Dp}PC<_{Dp}PM)})$
$-((CutSky(D^1, E) \bigcup CutSky(\{Dp\} \bigcup S, E)) \bigcup NewCompSky(D^2, E) =$
$(\{a, b, e, f, h\} \bigcup \{a, b, e\})$ - $(\{f\} \bigcup \{\}) \bigcup \{c\} = \{a, b, c, e, h\}$.

Preuve: (Théorème 2)
Nous considérons successivement les points qui sont disqualifiés du skyline et les points qui sont ajoutés au skyline. Soient $D^i \subseteq D$ et $D^i = D^{i-1} \bigcup \{d_i\}$. Soient Z le sous-espace de dimensions dynamiques de D et $Sky(D^i, E)$ le skyline associé au sous-espace D^i.

- Tout élément de $CutSky$ doit être disqualifié du skyline résultant.
  Si $p \in (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S))$ alors il existe un point $q \in Sky(D^i, E)$ tel que $(q =_{D^{i-1}} p$ et $q \prec_{d_i \bigcup S} p)$ ou $(q =_{d_i \bigcup S} p$ et $q \prec_{D^{i-1}} p)$.
  Cela signifie que $q \prec_{D^i} p$. Alors, p ne doit pas appartenir à $Sky(D^i, E)_{(Z,\wp)}$.
- Tout élément de $Sky(D^i, E)_{(Z,\wp)}$ doit appartenir à
  $\{ (Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)}) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)) \bigcup NewCompSky(D^i, E)\}$.
  Tout $p \in Sky(D^i, E)_{(Z,\wp)}$ est tel que :

  - soit il existe une dimension $d_j \in D^i$ telle que $\forall q \in E$, $p \preceq_{d_j} q$.
    Dans ce cas, $p \in (Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)})$
  - ou pour chaque $q \in E$, il existe une dimension $d_i \in Z$ telle que $p \prec_{d_i} q$. Dans ce cas, $p \in NewCompSky(D^i, E)$ et $p \notin (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)$

  Dans les deux cas p appartient à $Sky(D^i, E)_{(Z,\wp)}$

Le théorème 2 propose un schéma pour un calcul incrémental de skyline associés à des dimensions dynamiques. Dans ce qui suit, nous décrivons l'implémentation de notre proposition.

4.3.2 Implémentation de EC^2Sky

Dans cette sous-section, nous introduisons quelques définitions pour caractériser les points qui sont impliqués dans le calcul incrémental des skyline, et qui facilitent la description des algorithmes et de la structure de matérialisation de la méthode EC^2Sky. Afin d'assurer un calcul efficace et en ligne des skyline, nous proposons un compromis entre (i) *matérialiser* tous les points skyline pour toutes les préférences envisageables des utilisateurs et (ii) *calculer*, pour chaque requête utilisateur, les points skyline associés aux préférences formulées dans la requête. Notre approche comporte trois étapes :

1. calculer et stocker les skyline sur les dimensions statiques. Pour cela, nous pouvons adopter n'importe quel algorithme de calcul des skyline (voir sous-section 2.2.3) ;

2. pour chaque dimension dynamique, calculer et stocker les points skyline candidats (i.e. points susceptibles d'être skyline) pour chaque préférence de premier ordre possible ;
3. s'appuyer sur les informations stockées lors des deux étapes précédentes (i.e. 2 et 3) pour calculer les points skyline associés aux préférences de l'utilisateur sur les dimensions dynamiques progressivement introduites.

4.3.2.1 Les skyline associés aux dimensions statiques

Dans l'étape 1, nous calculons l'ensemble des skyline associés aux préférences statiques définies dans D. Deux concepts introduits par Wong et al. dans [WPFW09] nous seront utiles. En effet, ils décomposent l'ensemble $Sky(D, E)$, correspondant au skyline associé aux préférences statiques définies dans D et notées $\wp_\varnothing$, en deux sous-ensembles : le *global skyline set GSky(D,E)* et le *order-sensitive skyline set OsSky(D,E)*.

Les points appartenant à l'ensemble $GSky(D, E)$ restent skyline quelles que soient les préférences exprimées sur les dimensions de Z.

Définition 12 ***(Global skyline points)***
Le global skyline set associé à l'espace $D = S \bigcup Z$ et à l'ensemble de données E, est défini par
$GSky(D, E) = \{p \in Sky(D, E) \mid \forall\ q \in Sky(D, E), \nexists\ d_i \in Z, p =_S q \wedge p(d_i) \neq_{d_i} q(d_i)\}$

Certains points skyline sont qualifiés de order-sensitive car, dépendant des préférences associées aux dimensions dynamiques, ces points peuvent être skyline ou non. Notons que les points du global skyline set ne sont pas order-sensitive, et que les poins $CutSky$ sont à rechercher parmi les order sensitives.

Définition 13 ***(Order-sensitive skyline points)*** *L'ensemble skyline order-sensitive associé à l'espace D et à l'ensemble de données E, est défini par*
$OsSky(D, E) = \{p \in Sky(D, E) \mid p \notin GSky(D, E)\}$ *ou de manière équivalente* $OsSky(D, E) = Sky(D, E) - GSky(D, E)$.

Exemple 18 *Soient S = {Sn, Re} and Z = {Ts, Dp}. $GSky(D, E) = \{a, b, e\}$ et $OsSky(D, E) = \{c, d, f, h\}$, car tous les points skyline sont distincts.*

4.3.2.2 Les skyline associés aux dimensions dynamiques

Cette sous-section détaille l'étape 2 de notre approche. Dans cette étape, nous précalculons les informations utiles qui ne dépendent pas des préférences dynamiques exprimées par les utilisateurs. Pour chaque dimension dynamique d_i, nous introduisons *l'ensemble des skyline candidats* (CP_{d_i}), *l'ensemble des nouveaux skyline* ($NewSky_{(d_i,\wp_i^j)}$) et *l'ensemble des candidats compromis* ($CandComp_{(d_i,\wp_i^j)}$).

L'ensemble CP_{d_i} représente les points susceptibles de devenir skyline sur la dimension d_i. C'est le sous-ensemble des points de $OsSky(D, E)$:
(1) ayant sur $d_i \in Z$ une valeur différente de n'importe quel point de $GSky(D, E)$ qui

les dominent,
(2) ayant les mêmes valeurs sur les dimensions statiques et des valeurs différentes sur $d_i \in Z$.

Dans ce qui suit, $p \prec^j_{d_i \bigcup S} q$ indique que p domine q sur le sous-espace $d_i \bigcup S$ associé à la préférence de premier ordre $\wp^j_i$ de la dimension d_i.

Définition 14 ***(Points skyline candidats)*** *L'ensemble des skyline candidats de la dimension dynamique d_i, est défini par :*
$CP_{d_i} =$
$\{p \in OsSky(D,E) \mid \exists\ q \in GSky(D,E),\ q \prec_S p,\ p(d_i) \neq_{d_i} q(d_i)\} \bigcup$
$\{p \in OsSky(D,E) \mid \exists\ q \in OsSky(D,E),\ q =_S p,\ p(d_i) \neq_{d_i} q(d_i)\}$

Exemple 19 *Soit S = {Sn, Re} et d_i = {Ts}. On a $CP_{Ts} = \{c, d, f, h\}$.*

Afin de calculer les nouveaux skyline introduits par la nouvelle dimension d_i, il est suffisant de tester les points dans CP_{d_i} au lieu de tous les points non skyline. Cela peut réduire considérablement le nombre de tests de dominance.

$NewSky_{(d_i,\wp^j_i)}$ (Algorithme 1) représente l'ensemble des points dans CP_{d_i} qui sont préférés à l'ensemble des points dans $GSky(D,E)$ selon la préférence de premier ordre $\wp^j_i = v_j \prec_{d_i} *$ tel que $v_j \in dom(d_i)$. Intuitivement, les points NewSky sont équivalents aux points MaxSky (voir définition 5 dans le chapitre 2) associés à d_i selon la préférence $\wp^j_i$.

Définition 15 ***(Nouveaux points skyline)***
L'ensemble des nouveaux points skyline de la dimension dynamique d_i, est défini par :
$NewSky_{(d_i,\wp^j_i)} = \{p \in CP_{d_i} \mid \forall\ q \in GSky(D,E) \bigcup \{\ CP_{d_i} - p\},\ q \not\prec^j_{d_i \bigcup S} p\}$

Exemple 20 $NewSky_{Ts,H<_{Ts}*)} = \{c, d, f, h\}$.

Finalement, $CandComp_{(d_i,\wp^j_i)}$ (Algorithme 2) représente l'ensemble des points qui peuvent devenir des compromis skyline (i.e. des candidats compromis) lorsque l'on considère une nouvelle dimension. Cet ensemble est calculé pour chaque préférence de premier ordre $\wp^j_i$ sur d_i.

Définition 16 ***(Points candidats compromis)***
Soient $E' = (CP_{d_i}$ - $NewSky_{(d_i,\wp^j_i)})$ et $E'' = (GSky(D,E) \bigcup NewSky_{(d_i,\wp^j_i)})$.
Les points candidats compromis associés à la préférence $\wp^j_i$ est un ensemble de couples (p, Set_p) défini par :
$CandComp_{(d_i,\wp^j_i)} = \{(p, Set_p) \in E' \times \mathcal{P}(E'') \mid \forall q \in \mathcal{P}(E''),\ \exists d_k \in \{d_i\} \bigcup S,\ p \prec^j_{d_k} q\}$.

Exemple 21
$CandComp_{(Dp,PC<_{Dp}*)} = \{(f, \{a, b\})\}$ *où la notation $\{(f, \{a, b\})\}$ signifie que f appartient à $CandComp_{(Dp,PC<_{Dp}*)}$ car f domine a (resp. b) sur au moins une dimension de $\{Dp\} \bigcup S$ (ici Re (resp. Dp))).*

Algorithme 1 : Calcul de $NewSky_{(d_i,\wp_i^j)}$

input : d_i : une dimension dynamique, $\wp_i^j$: une préférence de premier ordre sur d_i, $GSky(D,E)$: global skyline set, CP_{d_i} : ensemble des skyline candidats sur d_i

output : $NewSky_{(d_i,\wp_i^j)}$

```
1 NewSky_(d_i,℘_i^j) ← ∅
2 pour chaque p ∈ CP_{d_i} faire
3     bool ← vrai
4     pour chaque q ∈ GSky(D,E) ⋃ { CP_{d_i} − p} faire
5         si q ≺^j_{d_i ⋃ S} p alors
6             bool← faux
7             quitter
8     si bool alors
9         NewSky_(d_i,℘_i^j) ← NewSky_(d_i,℘_i^j) ⋃{p}
```

4.3.3 Structure de $EC2Sky$

Voyons maintenant comment construire une structure EC^2Sky pour stocker efficacement toutes les informations pré-calculées. Notre objectif est d'éviter de construire une structure de données contenant toutes les combinaisons des préférences dynamiques sur toutes les dimensions tel que proposé dans [WFP+08]. Dans les sous-sections 4.3.2.1 et 4.3.2.2 et grâce au théorème 2, nous avons démontré que le skyline d'un sous-espace de dimensions étendu peut être calculé en tenant compte des préférences de premier ordre seulement. Nous proposons de stocker dans la structure EC^2Sky tous les ensembles $NewSky$ et $CandComp$ associés à chaque préférence de premier ordre de chaque dimension.

Pour chaque dimension d_i, nous calculons et stockons l'ensemble CP_{d_i}, et pour chaque préférence de premier ordre $\wp_i^j$ sur d_i, nous calculons et stockons les deux ensembles : $NewSky_{(d_i,\wp_i^j)}$ et $CandComp_{(d_i,\wp_i^j)}$ associés à la préférence de premier ordre j sur la dimension d_i. Les ensembles $NewSky_{(d_i,\wp_i^j)}$ et $CandComp_{(d_i,\wp_i^j)}$ associés à chaque préférence de premier ordre sur la dimension *Type de sol* ou sur la dimension *Durée prairie* sont présentées dans la table 4.2.

Nous évaluons maintenant la complexité spatiale de la structure EC^2Sky. Soit m le nombre de dimensions dynamiques et c la cardinalité maximale d'une dimension dynamique. La complexité spatiale de la structure EC^2Sky est donnée par :

$$\sum_{i=0}^{m}(c) = O(c.m)$$

Nous pouvons noter que la taille de la structure EC^2Sky est nettement plus petite que le nombre de préférences d'ordre n possibles donné par :

Algorithme 2 : Calcul de $CandComp_{(d_i,\wp_i^j)}$

input : d_i : une dimension dynamique, $\wp_i^j$: une préférence de premier ordre sur d_i, $NewSky_{(d_i,\wp_i^j)}$: points skyline introduits par d_i pour $\wp_i^j$, $GSky(D,E)$: global skyline set, CP_{d_i} : ensemble des skyline candidats sur d_i

output : $CandComp_{(d_i,\wp_i^j)}$

```
1 CandComp_(d_i,℘_i^j) ← ∅
2 pour chaque p ∈ {CP_di − NewSky_(d_i,℘_i^j)} faire
3     Set_p ← ∅
4     pour chaque q ∈ {GSky(D,E) ⋃ NewSky_(d_i,℘_i^j)} faire
5         pour chaque d_k ∈ {d_i} ⋃ S faire
6             si p ≺^j_dk q alors
7                 Set_p ← Set_p ⋃ q
8         CandComp_(d_i,℘_i^j) ← CandComp_(d_i,℘_i^j) ⋃ {(p, Set_p)}
```

$GSky = \{a,b,e\}$					
$CP_{Ts} = \{c,d,f,h\}$			$CP_{Dp} = \{f\}$		
$\wp = SG <_{Ts} *$	$\wp = SM <_{Ts} *$	$\wp = H <_{Ts} *$	$\wp = PC <_{Dp} *$	$\wp = PP <_{Dp} *$	$\wp = PM <_{Dp} *$
$NewSky_{\{Ts,\wp\}}$			$NewSky_{\{Dp,\wp\}}$		
$\{f,h\}$	$\{\}$	$\{c,d,f,h\}$	$\{\}$	$\{\}$	$\{f\}$
$CandComp_{\{Ts,\wp\}}$			$CandComp_{\{Dp,\wp\}}$		
$\{(c,\{a,b,e\}),$ $(d,\{a,b,e\})\}$	$\{(f,\{a\}),$ $(h,\{a\})\}$	$\{\}$	$\{(f,\{a,b\})\}$	$\{(f,\{a,e\})\}$	$\{\}$

TABLE 4.2 – Illustration d'une structure EC^2Sky avec deux dimensions dynamiques et trois préférences de premier ordre pour chaque dimension

$$(\sum_{i=0}^{c-1}(P_i(c)))^m = O((c.c!)^m)$$

Où $P_i(c)$ est le nombre de permutations de i éléments parmi c éléments. La complexité spatiale de la structure EC^2Sky est également beaucoup plus petite que la complexité spatiale de la structure $IPO\text{-}Tree$ donnée par :

$$\sum_{i=0}^{m}(c+1)^i = O(c^m)$$

Par exemple, lorsque $m = 3$ et $c = 40$, le nombre de préférences stockées dans la structure EC^2Sky est seulement égal à 123, tandis que dans la structure $IPO\text{-}Tree$, il

est égal à 70.644, et le nombre de toutes les préférences d'ordre n possibles ($n \in 1, .., c$) est égal à $4,1 * 10^9$. C'est 57.435 fois plus petit que l'$IPO\text{-}Tree$ et 714.502.572 plus petit que le nombre de toutes les préférences d'ordre n possibles. La différence est plus encore grande lorsque le nombre de dimensions m est plus élevé.

4.3.4 Évaluation de requêtes

Dans cette section, nous décrivons l'étape 3 de notre proposition (voir début de la sous-section 4.3.2). Les informations pré-calculées et stockées dans les étapes 1 et 2 sont utilisées dans l'étape 3 pour calculer, de manière interactive, le skyline associé aux préférences spécifiées dans la requête de l'utilisateur.

Algorithme 3 : Calcul de $CutSky(D^{i-1})$

input : $Sky(D^{i-1}, E)$, $Sky(D^i, E)$ et $Sky(d_i \bigcup S, E)$
output : $CutSky(D^{i-1})$

$CutSky(D^{i-1}) \leftarrow \varnothing$
pour chaque $p \in Sky(D^i, E)$ **faire**
  pour chaque $q \in Sky(D^{i-1}, E) \bigcup Sky(d_i \bigcup S, E)$ **faire**
    si $p =_{D^{i-1}} q$ *et* $q \prec_{d_i \bigcup S} p$ **alors**
      $CutSky(D^{i-1}) \leftarrow CutSky(D^{i-1}) \bigcup \{p\}$
      quitter

Algorithme 4 : Calcul de $NewCompSky(D^i, E)$

input : Structure EC^2Sky, $GSky(D, E)$: global skyline set
output : $NewCompSky(D^i, E)$

$CandCompSet = \bigcup_{i}\bigcup_{j} CandComp_{(d_i, \wp_i^j)}$
$NewCompSky(D^i, E) \leftarrow \varnothing$
pour chaque $p \in CandCompSet$ **faire**
  `// Calculer l'ensemble des points dominés par p sur au moins une dimension de` D^i
  $Domine(p) \leftarrow \{q | \exists\ d_i \in D^i, p \prec_{d_i} q\}$
  `// Sélectionner l'ensemble des points qui dominent les points skyline sur au moins une dimension`
  si $Domine(p) = Sky(D^{i-1}, E) \bigcup Sky(d_i \bigcup S, E) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)$ **alors**
    $NewCompSky(D^i, E) \leftarrow NewCompSky(D^i, E) \bigcup \{p\}$
Test de dominance sur tous les éléments de l'ensemble $NewCompSky(D^i, E)$

Une dimension avec des préférences dynamiques Dans un premier temps, nous considérons une seule dimension dynamique dans l'espace de dimensions D. Selon la requête de l'utilisateur, nous sommes confrontés à deux cas :

(i) *Requêtes avec des préférences de premier ordre* : pour calculer le skyline associé à une préférence de premier ordre $\wp_i^j$, nous utilisons les deux ensembles $GSky(D,E)$ et $NewSky_{(d_i,\wp_i^j)}$ stockés à l'étape 2, de la manière suivante : $Sky(d_i \bigcup S,E)_{(Z,\wp_i^j)} = GSky(D,E) \bigcup NewSky_{(d_i,\wp_i^j)}$. Rappelons que, lorsqu'il s'agit d'une seule dimension, il n'y a pas de points compromis ($CompSky = \varnothing$).

Exemple 22 *Nous utilisons la structure EC^2Sky décrite dans la table 4.2 pour illustrer les différentes étapes de l'évaluation d'une requête. Le skyline associé à la préférence $\wp = \{SG <_{Ts} *\}$ (stocké dans la structure EC^2Sky) est calculé comme suit :*
$Sky(\{Ts\} \bigcup S,E)_{(Z,SG<_{Ts}*)} = GSky(D,E) \bigcup NewSky_{\{Ts,SG<_{Ts}*\}}$.
Ainsi, $Sky(\{Ts\} \bigcup S,E)_{(Z,SG<_{Ts}*)} = \{a,b,e,f,h\}$ (i.e. skyline de $\wp$).

(ii) *Requêtes avec des préférences d'ordre n* : dans ce cas de figure, nous utilisons la *merging property* de Wong et al. [WFP+08] (voir théorème 1). Ceci est illustré par l'exemple suivant.

Exemple 23 *Le skyline associé à la préférence $\wp = \{SG <_{Ts} H <_{Ts} *\}$ peut être calculé à partir des points skyline relatifs aux préférences $\wp^1 = \{SG <_{Ts} *\}$ et $\wp^2 = \{H <_{Ts} *\}$ (stockés dans la structure EC^2Sky), comme suit :*
$Sky(\{Ts\} \bigcup S,E)_{(Z,SG<_{Ts}*)} = \{a,b,e,f,h\}$ *(i.e. skyline de $\wp^1$) (cf. exemple 22).*
De la même manière, $Sky(\{Ts\} \bigcup S,E)_{(Z,H<_{Ts}*)} = \{a,b,e,c,d,f,h\}$ (i.e. skyline de $\wp^2$).

Finalement, pour calculer le skyline associé à $\wp = \{SG <_{Ts} H <_{Ts} *\}$, nous utilisons la merging property :
$Sky(D,E)_{(Z,SG<_{Ts}H<_{Ts}*)} = (Sky(\{Ts\} \bigcup S,E)_{(Z,SG<_{Ts}*)} \bigcap$
$Sky(\{Ts\} \bigcup S,E)_{(Z,H<_{Ts}*)}) \bigcup PSky(D,E)_{(Z,SG<_{Ts}*)} =$
$(\{a,b,e,f,h\} \bigcap \{a,b,e,c,d,f,h\}) \bigcup \{f,h\} = \{a,b,e,f,h\}$

Plusieurs dimensions avec des préférences dynamiques Deuxièmement, nous considérons le cas de plusieurs dimensions dynamiques. Selon les définitions 9 et 11, certains points skyline (les points $CutSky$) peuvent être disqualifiés lorsqu'une nouvelle dimension est introduite, tandis que de nouveaux points skyline (les points $CompSky$) peuvent apparaître.

Le calcul de l'ensemble $CutSky$ est décrit par l'algorithme 3. Nous détaillons seulement comment calculer l'ensemble $CutSky(D^{i-1})$ étant donné que le calcul de $CutSky(d_i \bigcup S)$ est réalisé de la même manière.

Les points compromis skyline sont les candidats compromis qui sont devenus skyline. Le calcul de cet ensemble de points est décrit par l'algorithme 4.

Algorithme 5 : Construction de la structure EC^2Sky

input : E : ensemble de données, D : espace de dimensions de E, S : ensemble des dimensions statiques, Z : ensemble des dimensions dynamiques
output : EC^2Sky structure

`// Étape 1 : calcul des global et order-senstive skyline`
Calculer $GSky(D, E)$
Stocker $GSky(D, E)$ dans la structure EC^2Sky
Calculer $OsSky(D, E)$
Stocker $OsSky(D, E)$ dans la structure EC^2Sky
`// Étape 2 : calcul de la structure` EC^2Sky
pour chaque $d_i \in Z$ **faire**
    Calculer et stocker CP_{d_i} dans la structure EC^2Sky
    pour chaque $\wp_i^j \in \wp_i$ **faire**
        Calculer $NewSky_{(d_i,\wp_i^j)}$; `// Algorithme 1`
        Stocker $NewSky_{(d_i,\wp_i^j)}$ dans la structure EC^2Sky
        Calculer $CandComp_{(d_i,\wp_i^j)}$; `// Algorithme 2`
        Stocker $CandComp_{(d_i,\wp_i^j)}$ dans la structure EC^2Sky

Algorithme 6 : $EC^2Sky(Z, \wp)$

input : $Z, \wp$: requête skyline
output : $Sky(D, E)_{(Z,\wp)}$

`// Step 3 : calcul des points changeants`
$Sky(D^0, E)_{(Z,\wp)} \leftarrow GSky(D, E)$
pour $i \leftarrow 1$ **à** $m = |Z|$ **faire**
    si $\wp_i = \varnothing$ **alors**
        $Sky(d_i \bigcup S, E)_{(Z,\wp)} \leftarrow GSky(D, E) \bigcup CP_{d_i}$
    sinon
        si *estPrefPremierOrdre($\wp_i$)* **alors**
            $Sky(d_i \bigcup S, E)_{(Z,\wp)} \leftarrow GSky(D, E) \bigcup NewSky_{(d_i,\wp_i^j)}$
        sinon
            Utiliser la merging property
    Calculer $CutSky(D^{i-1})$ (resp. $CutSky(d_i \bigcup S)$)
    Calculer $NewCompSky(D^i, E)$; `// Algorithme 4`
    $Sky(D^i, E)_{(Z,\wp)} \leftarrow Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)} \bigcup NewCompSky(D^i, E) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S))$
$Sky(D, E)_{(Z,\wp)} \leftarrow Sky(D^m, E)_{(Z,\wp)}$

Nous sommes maintenant en mesure de détailler la méthode EC^2Sky. Les algorithmes 5 et 6 décrivent le processus général de EC^2Sky. L'algorithme 5 décrit les étapes nécessaires à la construction de la structure EC^2Sky. À la fin de l'étape 2 (Algorithme 5, lignes 8 et 11), nous calculons les ensembles de skyline stockés dans la structure EC^2Sky.

L'algorithme 6 est dédié à l'étape 3 i.e. le calcul des éléments changeants du skyline. Les ensembles *CompSky* (Algorithme 6, ligne 11) et l'union des *CutSky* (Algorithme 6, ligne 10) sont calculés.

Comme indiqué dans le théorème de calcul incrémental des skyline (Théorème 2), le skyline final est obtenu en éliminant tous les points $CutSky$ et en ajoutant tous les points $CompSky$ à l'union des skyline relatifs aux requêtes impliquant une préférence dynamique (Algorithme 6, ligne 12).

Exemple 24 *Le skyline associé aux préférences* $\wp = \{SG <_{Ts} H <_{Ts} *, PP <_{Dp} *\}$, *peut être calculé à partir du skyline associé aux préférences* $\wp_1 = \{SG <_{Ts} H <_{Ts} *\}$ *et* $\wp_2 = \{PP <_{Dp} *\}$.
Soient $D^1 = \{Sn, Re, Ts\}$ *et* $D^2 = \{Sn, Re, Ts, Dp\}$. *Le skyline associé à* $\wp_1$ *et* $\wp_2$ *est calculé de la même manière que dans l'exemple 23.*
$Sky(D^1, E)_{(Z,SG<_{Ts}H<_{Ts}*)} = \{a, b, e, f, h\}$ *et* $Sky(\{Dp\} \bigcup S, E)_{(Z,PP<_{Dp}*)} = \{a, b, e\}$.

Puisque nous avons deux dimensions dynamiques, nous calculons les ensembles $CutSky(D^1)$, $CutSky(\{Dp\} \bigcup S)$ *et* $NewCompSky(D^2, E)$. *Pour cet exemple,* $CutSky(D^1) = \varnothing$, $CutSky(\{Dp\} \bigcup S) = \varnothing$ *et* $NewCompSky(D^2, E) = \varnothing$.
Finalement,
$Sky(D^2, E)_{(Z,\wp)} = Sky(D^1, E)_{(Z,SG<_{Ts}H<_{Ts}*)} \bigcup Sky(\{Dp\} \bigcup S, E)_{(Z,PP<_{Dp}*)}$
$= \{a, b, e, f, h\}$.

Notre proposition fournit à l'utilisateur un moyen d'exprimer des préférences et de les changer sans être pénalisé par des temps d'attente trop longs. Les bonnes performances sont obtenues en ne stockant que le minimum d'informations requises afin d'autoriser des mises à jour simples et rapides.
L'évaluation expérimentale, présentée dans ce qui suit, souligne la pertinence de la solution proposée.

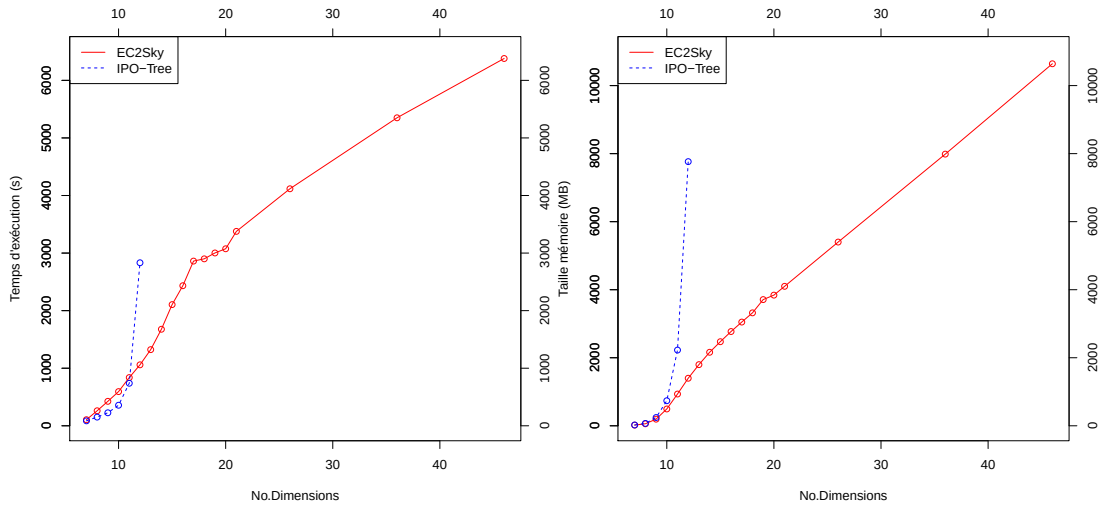
4.3.5 Évaluations expérimentales

Dans cette section, nous présentons une évaluation expérimentale de notre algorithme EC^2Sky sur des données synthétiques. L'algorithme EC^2Sky a été implémenté en *C/C++*. Nous avons conduit nos expérimentations sur un Intel Xeon CPU 3GHz et 16 GB de RAM sous une plateforme Linux. Pour les dimensions statiques, les données ont été produites par le générateur publié par les auteurs de [BKS01]. Trois types de jeux de données ont été générés : des données indépendantes, des données corrélées et des données non-corrélées. La description de ces jeux de données peut être trouvée dans [BKS01]. Comme dans [WFP+08], nous présentons seulement les résultats relatifs aux données non-corrélées. Les résultats relatifs aux données indépendantes et aux données

Paramètre	Valeur
Nb. de tuples	100K
Nb. de dimensions statiques	6
Nb. de dimensions dynamiques	4
Nb. de valeurs d'une dimension dynamqiue	4
Paramètre Zipfian θ	1

TABLE 4.3 – Valeurs par défaut

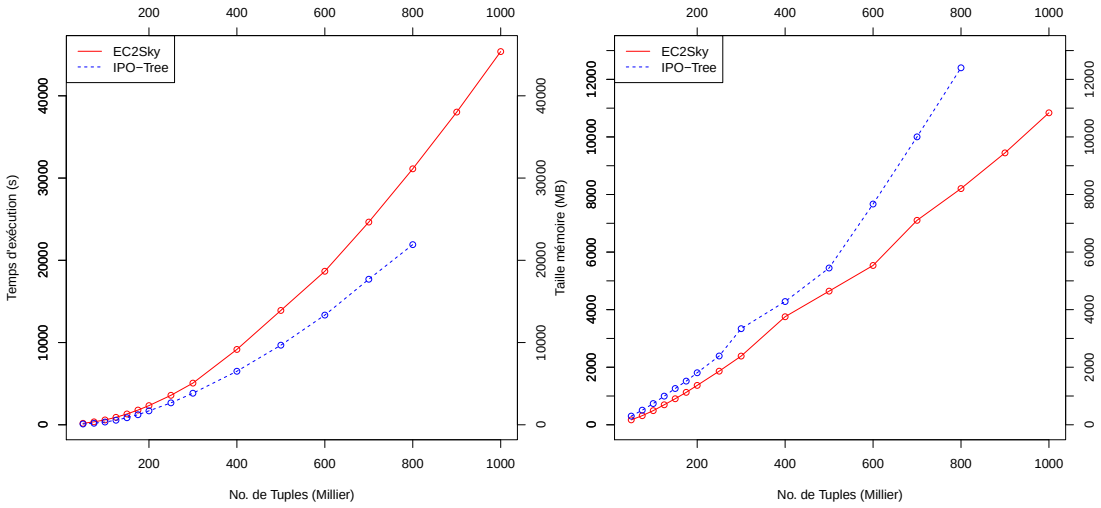
corrélées sont similaires, mais les temps d'exécution sont beaucoup plus courts pour les données corrélées. Les dimensions dynamiques ont été générées selon une distribution de Zipfian [Tre94]. Par défaut, nous avons initialisé le paramètre Zipfian θ à 1. Nous avons obtenus $1,000,000$ tuples avec 6 dimensions statiques. Nous avons fait varier le nombre de dimensions dynamiques de 1 à 40 et le nombre de valeurs de ces dimensions de 2 à 50. Nous avons choisi un template de requête de telle sorte que la valeur la plus fréquente de certaines dimensions dynamiques ait la plus haute priorité sur toutes les autres valeurs. Cela représente un critère qui devient plus difficile à gérer étant donné que le skyline tend à être plus grand.



(a) (b)

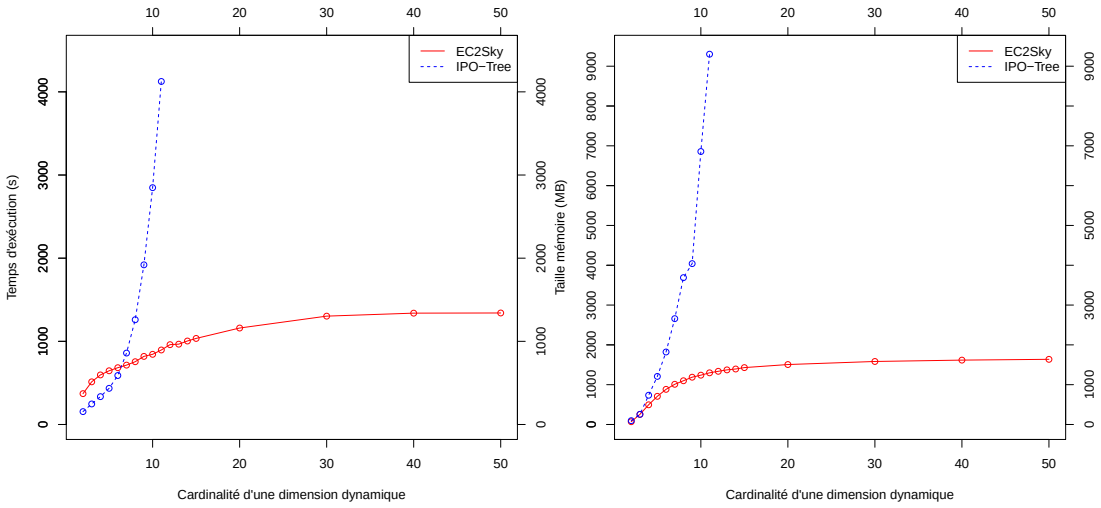
FIGURE 4.1 – Variation du nombre de dimensions

Dans ces expérimentations, nous comparons les performances de notre algorithme EC^2Sky avec l'algorithme *IPO-tree* implémenté par [WFP+08], et cela en terme de temps d'exécution et de stockage mémoire.



(a) (b)

FIGURE 4.2 – Variation de la taille de l'ensemble de données



(a) (b)

FIGURE 4.3 – Variation de la cardinalité des dimensions dynamiques

Variation du nombre de dimensions

Dans cette première expérimentation, le nombre de dimensions statiques est fixé à 6 et le nombre de dimensions dynamiques varie de 1 à 40. La Figure.4.1 montre que la taille

mémoire et le temps d'exécution de EC^2Sky et $IPO\text{-}tree$ augmentent avec le nombre de dimensions dynamiques. Cependant, le taux d'augmentation de $IPO\text{-}tree$ est supérieur à celui de EC^2Sky. Cela est dû à la complexité de l'arbre de préférences construit par $IPO\text{-}tree$. La structure de $IPO\text{-}tree$ contient plus de noeuds que celle de EC^2Sky, induisant un taux de stockage mémoire plus élevé. Au delà de 6 dimensions dynamiques, $IPO\text{-}tree$ déborde de la mémoire disponible. En effet, le nombre de noeuds de l'arbre $IPO\text{-}tree$ est de l'ordre de $O(c^m)$ (m étant le nombre de dimensions dynamiques et c la cardinalité d'une dimension dynamique), ce qui induit une évolution exponentielle du taux de stockage. La table construite par EC^2Sky a une taille de l'ordre de $O(c * m)$, ce qui induit certes une augmentation substantielle du taux de stockage mais qui évolue plus lentement que $IPO\text{-}tree$. Nous avons aussi étudié la variation du nombre de dimensions statiques lorsque le nombre de dimensions dynamiques est fixé à 4. Les résultats sont similaires à ceux de la Figure. 4.1.

Variation de la taille de l'ensemble de données

Dans cette expérimentation, le nombre de tuples de l'ensemble de données varie de 50.000 à 1.000.000. La Figure.4.2 montre que la taille mémoire et le temps d'exécution de EC^2Sky et $IPO\text{-}tree$ augmentent avec la taille de l'ensemble de données. Cela s'explique par le fait que la taille des informations stockées et analysées croit avec les données. Cependant, notre méthode est plus performante que $IPO\text{-}tree$. Au delà de 800.000 tuples, $IPO\text{-}tree$ déborde de la mémoire disponible. En effet, $IPO\text{-}tree$ stocke tous les skyline relatifs à toutes les combinaisons possibles des différentes préférences de premier ordre de toutes les dimensions dynamiques, tandis que EC^2Sky stocke seulement les points skyline correspondant aux préférences de premier ordre de chaque dimension dynamique. Les skyline des différentes combinaisons de préférences sont déduits à partir de simples opérations d'intersection et d'union.

Variation de la cardinalité des dimensions dynamiques

Nous avons fait varier la cardinalité des dimensions dynamiques de 2 à 50. La Figure.4.3 montre que la taille mémoire et le temps d'exécution de EC^2Sky et $IPO\text{-}tree$ augmentent avec le nombre de valeurs des dimensions. Une fois de plus, EC^2Sky est plus performant que $IPO\text{-}tree$. La taille de la structure $IPO\text{-}tree$ est exponentielle en $O(c^m)$. Ainsi, l'arbre construit par $IPO\text{-}tree$ devient plus complexe et plus volumineux avec l'augmentation de la cardinalité des dimensions (c). $IPO\text{-}tree$ déborde de la mémoire disponible pour une cardinalité au-dessus de 11. Nous pouvons également observer une augmentation significative du temps d'exécution associé.

4.4 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode efficace de calcul incrémental de skyline en présence de dimensions associées à des préférences dynamiques.

Nous avons étudié les préférences sur des valeurs de dimensions qui peuvent être exprimées par un ordre total ou partiel, avec une attention particulière portée aux points compromis qui sont essentiels pour une bonne prise de décision. Notre approche est basée sur une matérialisation des préférences de premier ordre, qui permet de répondre de manière efficace aux requêtes de type skyline en présence de préférences utilisateurs malgré de gros volumes de données. Cette méthode, EC^2Sky, respecte parfaitement tous les critères d'évaluation d'un algorithme de calcul de skyline (i.e. Efficacité, Progressivité, Préférences, Complétude et Correction) décrits précédemment dans la sous-section 2.2.2 du chapitre 2.

Les expérimentations réalisées dans cette étude soulignent l'efficacité et l'amélioration des performances de EC^2Sky comparé à *IPO-tree* [WFP+08].

La prise en compte des dimensions dynamiques ouvre plusieurs pistes de recherche prometteuses. Tout d'abord, afin de démontrer l'utilité de notre méthode, nous avons appliqué notre algorithme sur des jeux de données réels, notamment dans le cadre de l'analyse multidimensionnelle des résultats de simulations stockés dans l'entrepôt de données *N-Catch* décrit dans le chapitre 3. Ce travail fera l'objet de la première partie du chapitre 6.

Dans cette thèse, nous nous positionnons dans le cadre de bases de données multidimensionnelles et hiérarchiques (i.e. entrepôts de données). Nous pensons donc qu'il serait intéressant d'étudier les requêtes skyline dans un contexte de données hiérarchiques et agrégées. La démarche à adopter permettrait de rechercher les meilleurs compromis le long des axes des dimensions hiérarchiques. Ceci pose plusieurs problèmes, tout d'abord définir un calcul adapté au niveau de la hiérarchie explorée et ensuite permettre à l'utilisateur de naviguer tout au long des dimensions hiérarchiques (i.e. spécialiser/ généraliser) tout en assurant un calcul en ligne des skyline. Cette étude sera détaillée dans le prochain chapitre 5.

Chapitre 5

Requêtes skyline hiérarchiques

.

5.1 Introduction et motivations

Les requêtes skyline constituent un puissant outil d'analyse de données multidimensionnelles et d'aide à la décision. De nombreux travaux se sont intéressés à l'extraction de points skyline dans le contexte de bases de données multidimensionnelles, mais aucun de ces derniers n'a traité la problématique des skyline associés à des données agrégées lorsque les dimensions sont multiples et hiérarchiques. Dans le cadre de notre application agro-hydrologique (cf. chapitre 3), l'entrepôt de données *N-Catch* a été conçu selon un modèle multidimensionnel et hiérarchique (i.e. des hiérarchies peuvent être définies pour chaque dimension de *N-Catch*). Nous souhaitons coupler l'analyse skyline à l'analyse en ligne afin de permettre à l'utilisateur de naviguer vers les faits les plus intéressants du cube de données. En d'autres termes il s'agit de calculer les skyline en présence de dimensions hiérarchiques. Pour atteindre cet objectif, plusieurs verrous scientifiques et techniques sont à lever. Doit-on recalculer tous les skyline pour chaque niveau hiérarchique ? Peut-on les dériver à partir des skyline de niveau supérieur ou inférieur ? Peut-on étendre les algorithmes existants de calculs de skyline sur des dimensions hiérarchiques ?
Comme mentionné dans la sous-section 2.2.4 du chapitre 2, certains travaux récents [AWAA09, AWAEA08, WMZJ07, MI13] se sont intéressés à l'application des requêtes skyline sur des données agrégées. Ces travaux sont principalement axés sur l'optimisation de requêtes sollicitant les deux opérateurs *Skyline* et *Group-By* (opérateur d'agrégation). Cependant, aucun de ces travaux ne s'est intéressé à la prise en compte de la structure hiérarchique des dimensions dans le calcul des skyline. Pourtant, il serait pertinent pour un décideur de pouvoir naviguer le long des axes des dimensions hiérarchiques (i.e. spécialisant / généralisant) tout en assurant un calcul en ligne des points skyline correspondants. Dans le chapitre précédent, nous avons étudié le cas des dimensions dynamiques dans le contexte des requêtes skyline. Nous avons proposé la méthode EC^2Sky permettant à l'utilisateur de raffiner ses préférences de manière incrémentale

et d'assurer un calcul efficace des skyline associés. Dans ce chapitre, nous étudions le cas des dimensions hiérarchiques. À partir d'une préférence de base, notée $\wp^0$, et de l'ensemble skyline associé, nous souhaitons permettre à l'utilisateur de spécialiser ou de généraliser cette préférence et de dériver le skyline correspondant tout en respectant la structure hiérarchique des dimensions concernées. La spécialisation est un raffinement particulier : il est spécifique aux dimensions hiérarchiques et est soumis à des contraintes sémantiques liées à la structure hiérarchique de ces dimensions. La généralisation représente l'opération duale de la spécialisation.

Une solution naïve serait, soit de recalculer à chaque niveau hiérarchique l'ensemble des skyline, soit de stocker pour chaque niveau hiérarchique l'ensemble des skyline associés. Cependant, ces solutions sont très coûteuses en termes de temps de calcul et/ou de stockage mémoire dans un contexte de données volumineuses. Nous proposons donc une méthode $HSky$ (*Hierarchical Skyline queries*) permettant de réduire l'espace mémoire requis tout en assurant un calcul efficace et interactif des points skyline à tous les niveaux hiérarchiques des différentes dimensions. Nous mettons en évidence un ensemble de propriétés permettant la formalisation des relations hiérarchiques entre les préférences associées aux différentes dimensions. Ces propriétés permettent notamment aux utilisateurs de spécialiser ou de généraliser leurs préférences et d'extraire les points skyline associés de manière efficace.

Dans ce chapitre, nous présentons dans la section 5.2 les propriétés liées aux dimensions hiérarchiques et aux préférences associées à ces dernières dans le contexte du calcul des skyline. Dans la sous-section 5.2.4, nous détaillons les aspects formels ainsi que l'implémentation de notre proposition $HSky$ qui étend la recherche des points skyline aux dimensions hiérarchiques, et nous présentons les résultats de l'évaluation expérimentale réalisée sur des données synthétiques qui souligne la pertinence de la solution proposée.

5.2 Requêtes skyline associées à des dimensions hiérarchiques

Les différentes définitions dans cette section sont illustrées à partir de l'exemple de la table 5.1, décrivant un ensemble de parcelles selon la localisation (Loc), le rendement (Re) *kg/ha* et le cumul d'émissions d'azote du sol vers la nappe (Sn) *kgN/ha/année*. Une préférence de base $\wp^0_{d_i}$ est définie pour chaque dimension d_i de l'espace de dimensions D. Les valeurs des deux dimensions Re et Sn suivent la relation d'ordre $\leqslant$ qui spécifie qu'on préfère les parcelles ayant le taux de pollution nitrique i.e. Sn (resp. le rendement) le moins élevé (resp. le plus élevé). Les valeurs de la dimension Loc sont associées à la préférence de base suivante : $Bretagne <_{Loc} Epte$ et $Yeres <_{Loc} Normandie$. Les autres valeurs de la dimension Loc sont laissées non ordonnées.

Dans ce chapitre, nous considérons également des données avec des informations incomplètes (i.e. informations non spécifiées sur une dimension donnée). Par exemple, la parcelle e de la table 5.1 est décrite par la valeur ALL sur la dimension Loc. Cela signifie qu'il n'y a pas d'information sur l'endroit exact où la parcelle e se situe. De la même façon, la parcelle a de la table 5.1 est décrite par la valeur *Pays de Loire* sur la dimension Loc. Cela signifie que la valeur la plus détaillée et la plus précise dont nous

disposons concernant la localisation de la parcelle a est sa région (*Pays de Loire*). Pour les parcelles b, c, d et f, nous disposons des valeurs les plus précises du domaine de la dimension *Loc*.

Les préférences définies sur l'espace de dimensions D correspond à un ensemble de préférences binaires (ex. la préférence de la dimension *Loc* est composée de deux préférences binaires : ((*Bretagne*, *Epte*), (*Yeres*, *Normandie*)). Si le domaine des valeurs d'une dimension donnée est un ensemble indénombrable (ex. l'ensemble des réels) et totalement ordonné, alors l'ensemble des préférences binaires associées à ces valeurs ne peut être totalement listé. Par exemple, dans l'exemple de la table 5.1, l'ensemble des préférences binaires associées à la dimension *Re* (i.e. $\wp^0_{Re}$) ne peut être listé de façon exhaustive, et sera noté dans la suite par $\wp^0_{Re}$.

Dans la suite de ce chapitre, nous ne considérons que les fermetures transitives des préférences. Par exemple, à la préférence $\{(BRN, NOR), (NOR, PL)\}$ nous ferons correspondre sa fermeture transitive : $\{(BRN, NOR)\ ,\ (NOR, PL)\ ,\ (BRN, PL)\}$.

ID parcelle	Loc	Sn	Re
a	Pays de la Loire (PL)	16	200
b	Yeres (YRS)	24	500
c	Vilaine (VLN)	36	100
d	YAR	30	200
e	ALL	23	400
f	Epte (EPT)	30	300

TABLE 5.1 – Ensemble de parcelles

5.2.1 Formalisation des hiérarchies

Dans l'application *N-Catch*, le domaine de valeurs des dimensions peut être structuré sous forme de hiérarchies. La notion de hiérarchie de dimension est définie ci-dessous :

Définition 17 ***(Hiérarchie de dimension)*** *Une hiérarchie sur la dimension $d_i \in D$ est représentée par un graphe orienté acyclique dans lequel les noeuds représentent les valeurs du domaine de d_i, et les arcs représentent la relation d'inclusion ensembliste. La valeur la plus générale correspond au noeud le plus élevé de la hiérarchie (i.e. racine) et les feuilles correspondent aux valeurs les plus spécifiques.*

Une hiérarchie est associée à chaque dimension de l'espace de dimensions D.

Définition 18 *Soit $H_D = \{h_{d_1}, ..., h_{d_{|D|}}\}$ l'ensemble des hiérarchies associées aux dimensions de l'espace D.*

- *h_{d_i} représente la relation hiérarchique, éventuellement vide (i.e. graphe réduit au seul noeud ALL), entre les valeurs de la dimension $d_i \in D$,*

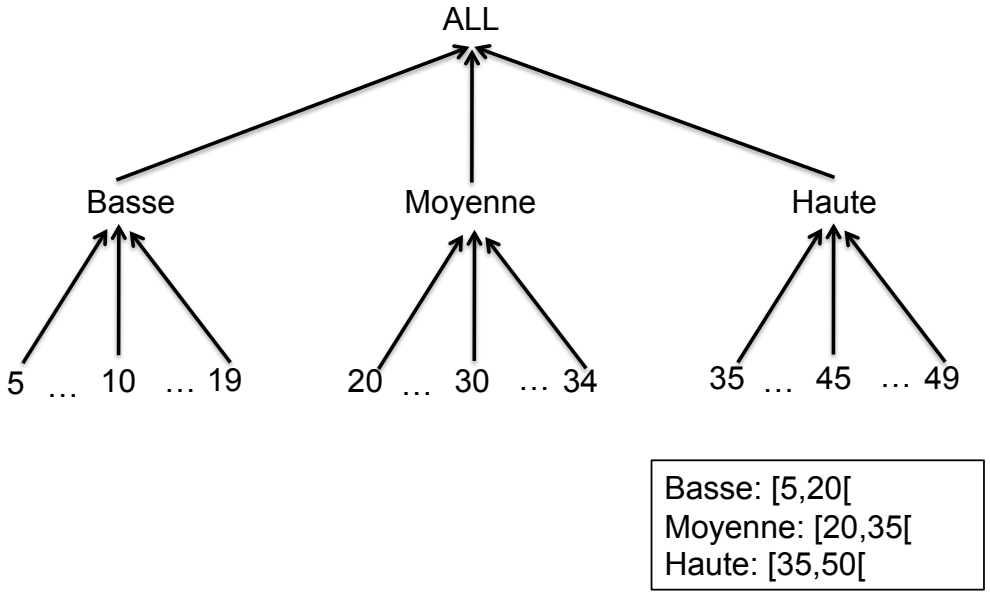
- h_{d_i} *est un graphe orienté acyclique,*
- *Pour chaque noeud* $n_j \in h_{d_i}$ *avec* $1 \leqslant j \leqslant |dom(d_i)|$, $label(n_j) = v_j$ *avec* $v_j \in dom(d_i)$.

$\widehat{v_j}$ *désigne l'ensemble des labels des ancêtres (directs ou indirects) de* n_j *dans la hiérarchie* h_{d_i}. $\check{v_j}$ *désigne l'ensemble des labels des descendants de* n_j.
Nous imposons que, pour chaque dimension d_i *de l'espace* D, *la valeur* ALL *appartient au domaine de valeurs de* d_i. *La valeur* ALL *correspond à la valeur la plus générale dans la hiérarchie d'une dimension quelconque. Les labels de tous les noeuds doivent apparaitre dans le domaine de la dimension correspondante.*

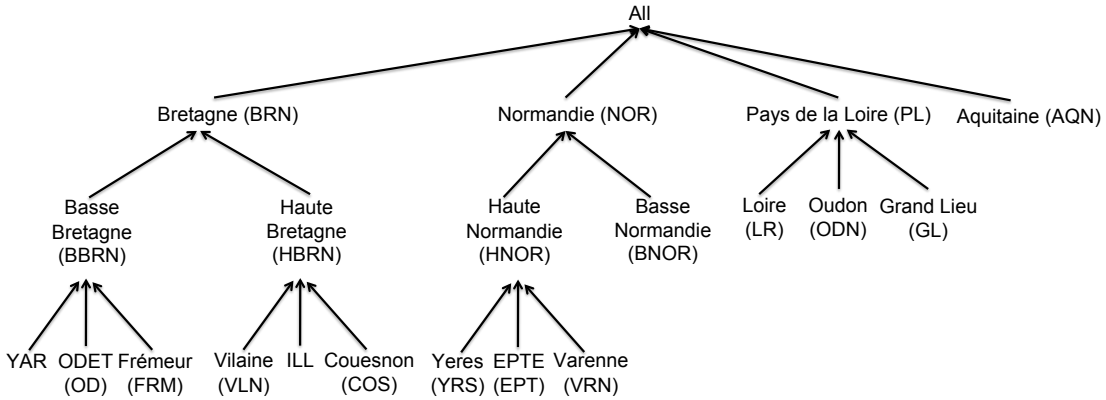
Exemple 25 *La Figure.5.1 donne des exemples de hiérarchies définies sur les dimensions Loc et Sn. Nous avons :*
$\widehat{VLN} = \{HBRN, BRN, ALL\}$, $\widetilde{NOR} = \{HNOR, BNOR, YRS, EPT, VRN\}$, $25 \in \widetilde{Moyenne}$.

La structure hiérarchique de la dimension Loc est représentée par trois niveaux hiérarchiques : la région (Bretagne, Normandie,...), la sous-région (Haute Bretagne, Basse Normandie,...) et le bassin versant agricole (Yar, Epte,...). Toutes les valeurs associées à ces niveaux hiérarchiques appartiennent au domaine de la dimension Loc. De même, pour les valeurs Basse, Moyenne, Haute et ALL qui sont ajoutées au domaine de valeurs de la dimension Sn.

La définition de hiérarchies sur les dimensions permet à l'utilisateur d'affiner ou de généraliser ses préférences lors du calcul des skyline. Prenons l'exemple de la dimension *Sn* décrite dans la table 5.1, nous remarquons qu'avant la structuration des valeurs de cette dimension en hiérarchie, l'utilisateur pouvait seulement formuler des préférences entre les valeurs individuelles (ex. une parcelle avec un taux de pollution nitrique de 16 est meilleure qu'une parcelle avec un taux de 30). Imaginons maintenant que l'utilisateur s'intéresse à des taux de pollution regroupés selon leur impact environnemental (ex. les taux inférieurs à 20 (i.e. bas) ont un faible impact, les taux entre 20 et 34 (i.e. moyens) ont un impact moyen et les taux supérieurs à 35 (i.e. hauts) ont un fort impact). L'utilisateur a besoin de formuler des préférences entre ces catégories de valeurs. La définition d'une hiérarchie sur la dimension *Sn* facilite la formulation de telles préférences et permet à l'utilisateur de poser des requêtes à différents niveaux hiérarchiques de la dimension. De la même manière, un utilisateur qui s'intéresse à l'étude du phénomène de proliférations d'algues vertes dans les bassins versants agricoles côtiers est susceptible de porter plus d'attention à la région de Bretagne (dimension *Loc*) qu'aux autres régions. L'utilisateur souhaitera alors accéder à un niveau plus précis des données relatives à la région de Bretagne afin de formuler ses préférences et d'extraire les points skyline associés. Il est alors intéressant de fournir à l'utilisateur un moyen d'affiner ses préférences à un niveau plus détaillé que la région (ex. niveau du bassin versant). La définition d'une hiérarchie sur la dimension *Loc* facilite de telles opérations.



(a) La hiérarchie de la dimension pollution nitrique Sol-Nappe *Sn*



(b) La hiérarchie de la dimension Localisation *Loc*

FIGURE 5.1

5.2.2 Relations hiérarchiques entre préférences

Dans cette sous-section, nous nous focalisons sur les nouvelles propriétés des préférences introduites par les dimensions hiérarchiques.

Propriété 2 ***(Consistance d'une préférence)***
Soit $\wp_{d_i} = \{(v_1, v_2), ..., (v_n, v_m)\}$ *une préférence sur la dimension hiérarchique* d_i*.*
La préférence $\wp_{d_i}$ *est consistante si et seulement si elle ne contient pas de préférence*

binaire ordonnant une valeur et un de ses ancêtres :
$\nexists(v_k, v_j) \in \wp_{d_i},\ k \neq j,\ (v_k \in \widehat{v_j} \vee v_j \in \widehat{v_k})$.

Exemple 26 *Prenons l'exemple de la table 5.1 où les hiérarchies de la Figure.5.1 sont associées aux dimensions Sn et Loc. la préférence* $\wp^0_{Loc} = \{(BRN, EPT), (YRS, NOR)\}$ *est inconsistante car* $\widehat{YRS} = \{NOR, HNOR, ALL\}$. *En effet, affirmer que "Le bassin versant du Yeres est préférable à la Normandie" n'a pas de sens puisqu'il existe un lien hiérarchique entre les deux valeurs "Yeres" et "Normandie" (i.e. Normandie est un ancêtre de Yeres). Pour rendre la préférence* $\wp^0_{Loc}$ *consistante, il faut supprimer la préférence binaire* (YRS, NOR) *ou changer la hiérarchie.*

La préférence $\{(YRS, PL)\}$ *est consistante car* PL *n'est pas un ancêtre de* YRS.

Dans la suite de ce chapitre, nous considérons seulement des préférences consistantes.

Définition 19 ***(Fermeture hiérarchique)***
Soit $\wp_{d_i}$ *une préférence associée à la dimension hiérarchique* d_i *de l'espace de dimensions* D.
La fermeture hiérarchique de la préférence $\wp_{d_i}$ *est définie par l'ensemble des préférences binaires suivant :*
$\{(v_p, v_q) \mid \exists\ (v_n, v_m) \in \wp_{d_i},\ (v_p \in \widetilde{v_n} \vee v_p = v_n) \wedge (v_q \in \widetilde{v_m} \vee v_q = v_m)\}$.

Exemple 27 *Soit* $\wp_{Loc} = \{(BRN, EPT)\}$.
La fermeture hiérarchique de $\wp_{Loc} = \{(BRN, EPT), (BBRN, EPT),$
$(HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT),$
$(COS, EPT)\}$.

Propriété 3 *La fermeture hiérarchique d'une préférence* $\wp$ *est consistante ssi* $\wp$ *est consistante.*

Dans la suite de ce chapitre, nous ne considérons que les fermetures hiérarchiques des préférences.

Comme mentionné précédemment, les dimensions de l'espace D sont associées à une préférence de base $\wp^0$. Pour chaque dimension $d_i \in D$, la préférence de base détermine un ordre sur les valeurs du domaine de d_i qui intéressent l'utilisateur. Lorsqu'une hiérarchie est associée à une dimension d_i, l'utilisateur a la possibilité de naviguer dans la hiérarchie de valeurs et ainsi de spécialiser ou de généraliser les valeurs ordonnées dans la préférence de base en suivant les liens hiérarchiques.

La spécialisation d'une préférence correspond à l'association d'un ordre sur les valeurs d'un niveau plus détaillé de la hiérarchie de la dimension concernée.

Dans un premier temps, nous donnons à l'utilisateur la possibilité de spécialiser uniquement les valeurs explicitement citées dans la préférence de base. La spécialisation d'une valeur d'une préférence correspond à l'association d'un ordre (total ou partiel) sur tous les descendants directs de la valeur en question. Ces descendants doivent être

initialement non ordonnés. Si un ordre est déjà défini sur une partie ou sur la totalité de ses descendants, la complétion au même niveau de détail de cet ordre n'est pas considérée comme une spécialisation mais comme un raffinement (cf. définition 8, chapitre 4).

Par exemple, à partir des valeurs explicitement ordonnées dans la préférence de base $\wp_{Loc} = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\}$, nous pouvons désigner l'ensemble des valeurs du domaine de *Loc* qui peuvent être spécialisées (cf. les valeurs en vert de la Figure.5.2) et celles qui ne peuvent pas l'être (cf. les valeurs en rouge de la Figure.5.2). Ceci permet de tracer une frontière entre ces deux types de valeurs (cf. Figure.5.2).

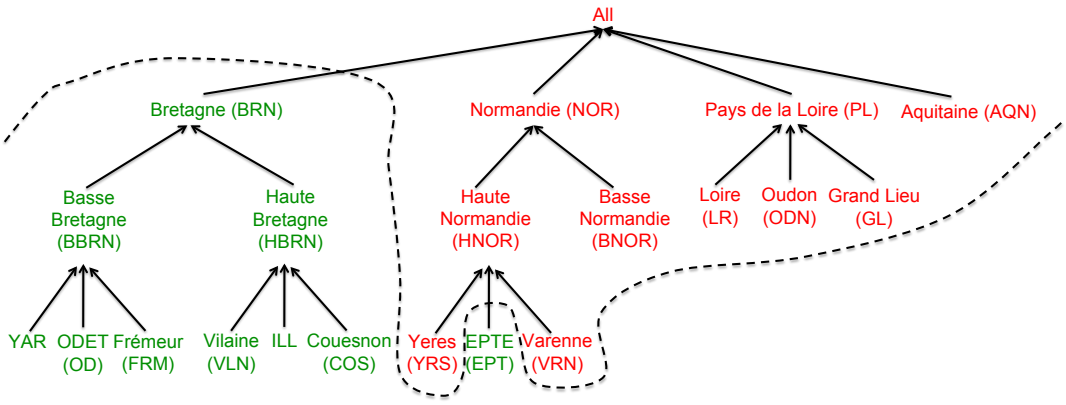


FIGURE 5.2 – Frontière explicite de spécialisation/généralisation pour la préférence de base $\wp^0_{Loc}$

Exemple 28 *Soient $\wp^0_{Loc}$ et $\wp'_{Loc}$ deux préférences associées à la dimension hiérarchique Loc.*
$\wp^0_{Loc} = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\}$ *(fermeture hiérarchique de {(BRN,EPT)}) et* $\wp'_{Loc} = \wp^0_{Loc} \bigcup \{(YAR, VLN)\}$.
$\wp'_{Loc}$ *est un raffinement de* $\wp^0_{Loc}$ *car* $\wp'_{Loc}$ *étend* $\wp^0_{Loc}$ *avec l'ensemble de préférences binaires {(YAR,VLN)}.*
$\wp'_{Loc}$ *est une spécialisation de* $\wp^0_{Loc}$ *car les valeurs de* $\{(YAR, VLN)\}$ *sont des descendants de BRN dans* h_{Loc}.

Considérons maintenant l'exemple suivant :
$\wp^0_{Loc} = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\}$ *et*
$\wp'_{Loc} = \wp^0_{Loc} \bigcup \{(VLN, GL)\}$.

$\wp'_{Loc}$ *est un raffinement de* $\wp^0_{Loc}$ *mais* $\wp'_{Loc}$ *n'est pas une spécialisation de* $\wp^0_{Loc}$. *En*

effet, la valeur GL citée dans $\wp'_{Loc}$ n'est pas la spécialisation d'une des valeurs citées dans $\wp^0_{Loc}$ (i.e. valeurs en vert dans la Figure.5.2).

Nous relaxons maintenant la contrainte précédente en étendant la possibilité de spécialiser des éléments n'apparaissant pas explicitement dans la préférence de base.

Lorsque la préférence associée à une dimension hiérarchique donnée d_i correspond à un ordre partiel, certaines valeurs du domaine de d_i restent non ordonnées par rapport aux valeurs citées explicitement dans la préférence en question. Dans le contexte des requêtes skyline, un point ayant une valeur non ordonnée sur la dimension d_i ne peut être dominé par aucun autre point sur cette dimension. À partir de ce constat, il est intéressant de donner la possibilité à l'utilisateur de spécialiser ces valeurs qui sont susceptibles de l'intéresser lors du calcul des skyline. La Figure.5.3 décrit l'évolution de la nouvelle frontière qui sépare les valeurs spécialisables des valeurs qui ne le sont pas. Les valeurs en bleu correspondent aux nouvelles valeurs dont les descendants peuvent être associés à une préférence supplémentaire (spécialisations des valeurs indifférentes).

Exemple 29 *Soient $\wp^0_{Loc}$ et $\wp_{Loc}$ deux préférences associées à la dimension hiérarchique Loc.*
$\wp^0_{Loc} = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\}$ *(fermeture hiérarchique de {(BRN,EPT)}) et $\wp_{Loc} = \wp^0_{Loc} \bigcup \{(LR, GL)\}$.*
$\wp^0_{Loc}$ est une généralisation de $\wp'_{Loc}$ par rapport à h_{Loc}. En effet, l'ancêtre de GL et LR (i.e. PL) n'apparaît pas dans la préférence $\wp^0_{Loc}$. Cela signifie que la valeur PL n'est pas ordonnée explicitement dans $\wp^0_{Loc}$ par rapport aux autres valeurs (i.e. BRN et NOR).

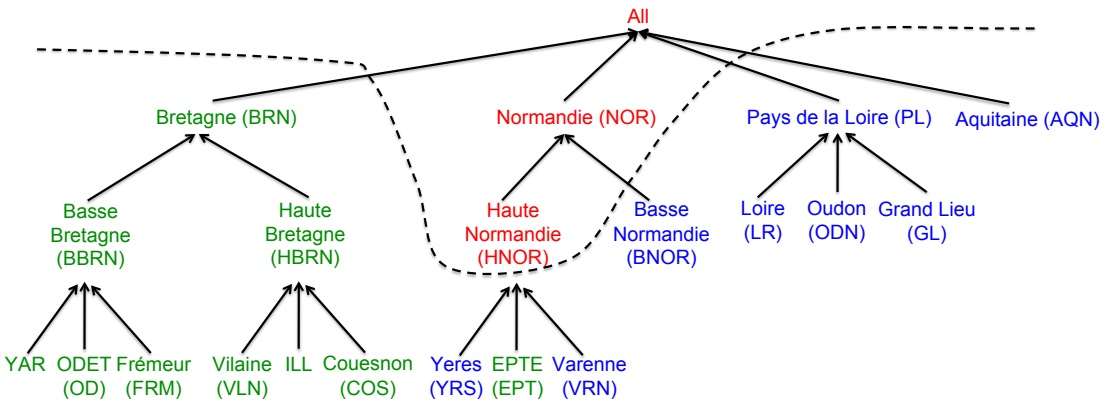


FIGURE 5.3 – Frontière implicite de spécialisation/généralisation

La possibilité de spécialiser une préférence implique implicitement la possibilité de la généraliser. La généralisation d'une préférence étant l'opération duale de la spécialisation d'une préférence, elle consiste en l'ajout de l'indifférence entre les valeurs explicitement ou implicitement ordonnées de la préférence en question. Autrement dit, il s'agit

de relaxer un ordre exprimé sur les descendants d'une valeur citée dans la préférence en question.

Exemple 30 *Soient* $\wp^0_{Loc} = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\}$ *et* $\wp^1_{Loc} = \wp^0_{Loc} \bigcup \{(VLN, YAR)\}$ *deux préférences associées à la dimension Loc.*
$\wp^0_{Loc}$ *est une généralisation de* $\wp^1_{Loc}$ *par rapport à* h_{Loc}*, car* $\widehat{YAR} = \widehat{VLN} = BRN$.
Prenons un autre exemple avec $\wp^0_{Loc}$ *et* $\wp^2_{Loc} = \wp^0_{Loc} \bigcup \{(GL, ODN)\}$.
$\wp^0_{Loc}$ *est une généralisation de* $\wp^2_{Loc}$ *par rapport à* h_{Loc}*. En effet, l'ancêtre de GL et ODN (i.e. PL) n'apparaît pas dans la préférence* $\wp^0_{Loc}$*. Cela signifie que la valeur PL n'est pas ordonnée dans* $\wp^0_{Loc}$ *par rapport aux autres valeurs (i.e. BRN et NOR).*
Prenons un dernier exemple $\wp^1_{Loc}$ *et* $\wp^3_{Loc} = \wp^1_{Loc} \bigcup \{(YAR, OD)\}$.
$\wp^1_{Loc}$ *n'est pas une généralisation de* $\wp^3_{Loc}$*.* $\wp^3_{Loc}$ *étend la préférence* $\wp^1_{Loc}$ *et est un raffinement de* $\wp^1_{Loc}$ *car elle supprime de l'indifférence implicite entre YAR et OD.*

La définition 20 définit formellement les opérations de spécialisation/généralisation :

Définition 20 ***(Spécialisation/Généralisation d'une préférence)***
Soient $\wp_{d_i} = \{(v_1, v_2), ..., (v_n, v_m)\}$ *et* $\wp'_{d_i} = \{(v'_1, v'_2), ..., (v'_p, v'_q)\}$ *deux préférences associées à la dimension hiérarchique* d_i*.*

- $\wp'_{d_i}$ *est une spécialisation de* $\wp_{d_i}$ *(notée* $\wp_{d_i} \subset_h \wp'_{d_i}$*) si :*
  1. $\wp_{d_i} \subset \wp'_{d_i}$ *i.e.* $\wp'_{d_i}$ *est un raffinement de* $\wp_{d_i}$ *(cf. définition 8, chapitre 4),*
  2. $\forall (v'_i, v'_j) \in \wp'_{d_i}$ *:*
     - *soit (a) :* $(v'_i, v'_j) \in \wp_{d_i}$,
     - *soit (b) :* $\exists (v_k, v_m) \in \wp_{d_i}$, $(v'_i \in \breve{v_k} \wedge v'_j \in \breve{v_k}) \vee (v'_i \in \widetilde{v_m} \wedge v'_j \in \widetilde{v_m})$,
     - *soit (c) :* $\widehat{v'_i} \cap \widehat{v'_j} \neq \emptyset \wedge \forall v \in \widehat{v'_i} \cap \widehat{v'_j}$, v *n'est pas ordonnée dans* $\wp_{d_i}$*.*

     Et il existe au moins un couple (v'_i, v'_j) *qui vérifié la propriété (b).*
- $\wp_{d_i}$ *est une généralisation de* $\wp'_{d_i}$ *si* $\wp'_{d_i}$ *est une spécialisation de* $\wp_{d_i}$*.*
- $\wp_{d_i}$ *et* $\wp'_{d_i}$ *sont incomparables s'il n'existe aucune relation hiérarchique entre elles* $(\neg(\wp_{d_i} \subset_h \wp'_{d_i}) \wedge \neg(\wp'_{d_i} \subset_h \wp_{d_i}))$*,*

Maintenant, nous étendons la définition de spécialisation/généralisation d'une préférence à plusieurs dimensions hiérarchiques.

Définition 21 *Soient* $\wp = \bigcup_{d_i \in D} \wp_{d_i}$ *et* $\wp' = \bigcup_{d_i \in D} \wp'_{d_i}$ *deux préférences associées à l'espace de dimensions D.*

- $\wp'$ *est une spécialisation de* $\wp$ *(i.e.* $\wp$ *est une généralisation de* $\wp'$*) ssi :*
  1. $\forall d_i \in D$, $\wp_{d_i} \subset_h \wp'_{d_i} \vee \wp_{d_i} = \wp'_{d_i}$,
  2. $\exists d_i \in D$, $\wp_{d_i} \subset_h \wp'_{d_i}$.

Toutes les spécialisations et généralisations d'une préférence associée à un ensemble de dimensions constituent un ensemble partiellement ordonné où l'ordre est déterminé par la relation de spécialisation/généralisation. Par exemple, toutes les spécialisations

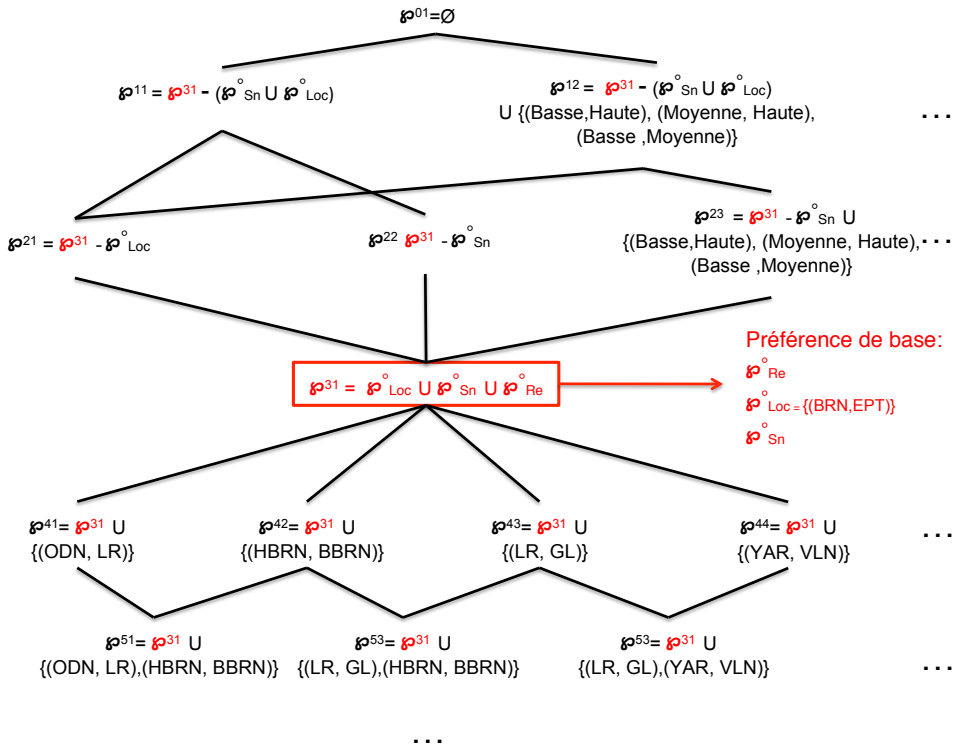


FIGURE 5.4 – Structure de spécialisation/généralisation

et généralisations de la préférence de base $\wp^0$ définie sur les dimensions de la table 5.1, induisent la structure hiérarchique de la Figure.5.4. Chaque noeud de cette structure est associé à une spécialisation ou à une généralisation de la préférence de base $\wp^0$.

La notation $\wp^{kl}$, avec k un numéro de niveau dans la structure hiérarchique et l un numéro séquentiel associé à un noeud de niveau k, est parfois utilisée dans certaines figures afin de les rendre plus lisibles.

D'après la définition 20 définie ci-dessus, la spécialisation d'une préférence implique le raffinement de cette dernière, mais l'inverse n'est pas vérifié. Nous pouvons alors déduire un corollaire de la propriété 1 de monotonie du raffinement d'une préférence (voir section 4.2 chapitre 4).

Corollaire 1 ***(Monotonie hiérarchique)*** *Soient $\wp$ et $\wp'$ deux préférences sur l'espace de dimensions D. Si $\wp'$ est une spécialisation de $\wp$ ($\wp \subset_h \wp'$), alors $\wp'$ est aussi un raffinement de $\wp$, et par conséquent, $Sky(D,E)_{\wp'} \subseteq Sky(D,E)_{\wp}$.*

Exemple 31 *Soient $\wp^0 = \{(BRN,EPT), (BBRN,EPT), (HBRN,EPT), (YAR,EPT), (OD,EPT), (FRM,EPT), (ILL,EPT), (VLN,EPT), (COS,EPT)\} \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$ la préférence de base définie sur l'espace D, et $\wp' = \wp^0_{Loc} \bigcup \{(YAR,VLN)\} \wp^0_{Sn} \bigcup \wp^0_{Re}$ une préférence définie sur ce même espace.*

$\wp'$ est une spécialisation de $\wp^0$.
$Sky(D,E)_{\wp^0} = \{a, b, e, c, d, f\}$ et $Sky(D,E)_{\wp'} = \{a, b, d, e, f\}$.
Nous avons bien $Sky(D,E)_{\wp'} \subseteq Sky(D,E)_{\wp^0}$.

5.2.3 Navigation dans la structure de spécialisation/généralisation

La spécialisation d'une préférence donnée $\wp$ implique la recherche d'un noeud descendant (direct ou indirect) de la préférence $\wp$ dans la structure de spécialisation / généralisation. La spécialisation de la préférence $\wp$ désigne l'ajout d'une préférence (i.e. ensemble de préférences binaires) aux valeurs d'un niveau plus détaillé qui n'en comporte pas. Une fois la spécialisation effectuée, si nous souhaitons compléter au même niveau de détail la préférence rajoutée, nous devons d'abord la supprimer (i.e. généraliser $\wp$) pour revenir à la préférence initiale $\wp$, et ensuite la spécialiser en rajoutant la nouvelle préférence souhaitée. Autrement dit, nous remontons d'un niveau ou plusieurs dans la structure de spécialisation/généralisation pour naviguer vers le même ou un autre noeud descendant. Dans l'exemple 30, $\wp^0_{Loc}$ peut être spécialisée en $\wp^1_{Loc}$ ou en $\wp^2_{Loc}$. Par contre, si nous souhaitons spécialiser $\wp^1_{Loc}$ en $\wp^2_{Loc}$ (resp. $\wp^2_{Loc}$ en $\wp^1_{Loc}$), il faut d'abord généraliser $\wp^1_{Loc}$ en $\wp^0_{Loc}$ (resp. $\wp^2_{Loc}$ en $\wp^0_{Loc}$), et ensuite spécialiser $\wp^0_{Loc}$ en $\wp^2_{Loc}$ (resp. en $\wp^1_{Loc}$), car $\wp^1_{Loc}$ n'est ni une spécialisation ni une généralisation de $\wp^2_{Loc}$.

La généralisation d'une préférence $\wp$ consiste à rajouter de l'indifférence sur un ensemble de valeurs appartenant à la même valeur ancêtre dans la hiérarchie d'une dimension. Dans la structure de spécialisation/généralisation, cela revient à rechercher un noeud ancêtre (direct ou indirect) de la préférence $\wp$.

Notre objectif est de minimiser le nombre de tests de dominance pour calculer efficacement les skyline lors de la navigation sur les dimensions hiérarchiques. Nous souhaitons donc caractériser les points skyline qui restent skyline et ceux qui ne le sont plus, lors de la spécialisation (drill-down) ou de la généralisation (roll-up) des préférences associées aux dimensions hiérarchiques de l'entrepôt de données *N-Catch*.

La propriété de monotonie hiérarchique contribue considérablement à l'analyse et au calcul des points skyline en présence de préférences associées à des dimensions hiérarchiques. En effet, cette propriété indique que tout point skyline associé à une préférence donnée reste skyline lorsque nous considérons une généralisation de cette préférence.

5.2.4 Algorithme *HSky*

Dans cette sous-section, nous définissons les requêtes skyline hiérarchiques sur l'espace D, comportant des dimensions hiérarchiques.

Afin d'assurer un calcul efficace et en ligne de l'ensemble skyline dans un contexte multidimensionnel et hiérarchique, nous construisons une structure de matérialisation qui sera détaillée dans la suite. Nous proposons un compromis entre (i) *matérialiser* tous les points skyline de toutes les préférences hiérarchiques (i.e. spécialisations/généralisations) associées à la préférence de base $\wp^0$, et (ii) *calculer*, pour chaque requête utilisateur, les points skyline associés aux préférences hiérarchiques formulées dans la requête.

Dans la suite, nous présentons les spécifications de la structure de matérialisation, nommée $HSky$, ainsi que les algorithmes impliqués dans le calcul des requêtes skyline hiérarchiques.

5.2.4.1 Calcul des requêtes skyline hiérarchiques

Dans le but de calculer les points skyline pour une requête utilisateur associée à une généralisation quelconque $\wp$ de la préférence $\wp'$, nous introduisons l'ensemble de points *hierarchical skyline* $HNSky(D,E)_{(\wp',\wp)}$, où $\wp$ est un ancêtre direct de $\wp'$ dans la structure de spécialisation/généralisation. Cet ensemble contient les nouveaux points skyline introduits par la généralisation de $\wp'$ en $\wp$ (il regroupe les points qui sont disqualifiées lors de la spécialisation de $\wp$ en $\wp'$). L'utilisation de cet ensemble permettra de limiter les tests de dominance lors de la construction de la structure $HSky$. Partant d'une préférence de base et de son ensemble skyline, et grâce à la propriété de monotonie hiérarchique, nous pouvons calculer le skyline des différentes spécialisations ou généralisations de la préférence de base à partir du skyline qui lui est associé.

Définition 22 ***(HNSky : Hierarchical New Skyline)*** *Soient $\wp$ et $\wp'$ deux préférences associées à D avec $\wp'$ une spécialisation directe de $\wp$, $Sky(D,E)_{\wp'}$ le skyline associé à $\wp'$ et $Sky(D,E)_{\wp}$ le skyline associé à $\wp$.*
L'ensemble des points de $HNSky_{(\wp,\wp')}$ est défini par :
$HNSky(D,E)_{(\wp,\wp')} = \{p \in Sky(D,E)_{\wp} | p \notin Sky(D,E)_{\wp'}\}$.

Exemple 32 *Soient $\wp^0 = \{(BRN,EPT),(BBRN,EPT),(HBRN,EPT),(YAR,EPT),(OD,EPT),(FRM,EPT),(ILL,EPT),(VLN,EPT),(COS,EPT)\} \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$ la préférence de base définie sur l'espace D, et $\wp' = \wp^0 \bigcup \{(YAR,VLN)\}$ une préférence définie sur ce même espace.*
$\wp'$ est une spécialisation directe de $\wp^0$.
$Sky(D,E)_{\wp^0} = \{a,b,e,c,d,f\}$ et $Sky(D,E)_{\wp'} = \{a,b,d,e,f\}$.
Par conséquent, $HNSky(D,E)_{(\wp^0,\wp')} = \{c\}$.

Supposons que nous souhaitions naviguer dans la structure des spécialisations / généralisations des préférences, tout en assurant un calcul en ligne des skyline. Le théorème suivant indique que lors de la généralisation de la préférence $\wp'$ en $\wp$, le skyline associé à $\wp$ peut être obtenu en ajoutant au skyline associé à la préférence $\wp'$ les nouveaux points skyline (i.e. l'ensemble $HNSky$) introduits par la généralisation, comme suit :

Théorème 3 ***(HSky : Hierarchical Skyline)*** *Soient E un ensemble de données à $|D|$ dimensions, $\wp$ et $\wp'$ deux préférences associées à D telle que $\wp \subset_h \wp'$.*
Soit $Sky(D,E)_{\wp'}$ le skyline associé à la préférence $\wp'$.
$Sky(D,E)_{\wp} = Sky(D,E)_{\wp'} \bigcup HNSky(D,E)_{(\wp,\wp')}$.

Preuve:
Soient E un ensemble de données à $|D|$ dimensions, $\wp$ et $\wp'$ deux préférences associées à D tel que $\wp \subset_h \wp'$.

Soit $Sky(D,E)_{\wp'}$ le skyline associé à la préférence $\wp'$. Nous avons seulement besoin de prouver les deux cas suivants :

1. Chaque point skyline de $Sky(D,E)_{\wp'} \bigcup HNSky(D,E)_{(\wp,\wp')}$ appartient à $Sky(D,E)_{\wp}$.

   – à partir de la propriété 1 : Comme $\wp \subset_h \wp'$, $Sky(D,E)_{\wp'} \subseteq Sky(D,E)_{\wp}$. Alors, pour chaque $p \in Sky(D,E)_{\wp'}$, $p \in Sky(D,E)_{\wp}$;
   – ou à partir de la définition 22 : Comme $\wp \subset_h \wp'$, $p \in HNSky(D,E)_{(\wp,\wp')}$ signifie que $p \in Sky(D,E)_{\wp}$.

   Dans tous les cas p appartient à $Sky(D,E)_{\wp}$.

2. Chaque point skyline de $Sky(D,E)_{\wp}$ appartient à $Sky(D,E)_{\wp'} \bigcup HNSky(D,E)_{(\wp,\wp')}$.

   À partir de la propriété 1, pour chaque $p \in Sky(D,E)_{\wp}$, deux cas se présentent :
   – p n'est pas un élément de $Sky(D,E)_{\wp'}$. Alors, à partir de la définition 22, $p \in HNSky(D,E)_{(\wp,\wp')}$;

   – $p \in Sky(D,E)_{\wp'}$. À partir de la propriété 1 : comme $\wp \subset_h \wp'$, $Sky(D,E)_{\wp'} \subseteq Sky(D,E)_{\wp}$. Alors, pour chaque $p \in Sky(D,E)_{\wp'}$, $p \in Sky(D,E)_{\wp}$;
   Dans tous les cas p appartient à $Sky(D,E)_{\wp'} \bigcup HNSky(D,E)_{(\wp,\wp')}$.

Exemple 33 ***Illustration du théorème 3*** *Soient* $\wp^0 = \{(BRN,EPT),(BBRN,EPT),(HBRN,EPT),(YAR,EPT),(OD,EPT),(FRM,EPT),(ILL,EPT),(VLN,EPT),(COS,EPT)\} \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$ *la préférence de base définie sur l'espace D, et* $\wp' = \wp^0_{Loc} \bigcup \{(YAR,VLN)\}\ \wp^0_{Sn} \bigcup \wp^0_{Re}$ *une préférence définie sur ce même espace.*
$\wp^0$ *est une généralisation directe de* $\wp'$.
$Sky(D,E)_{\wp^0} = Sky(D,E)_{\wp'} \bigcup HNSky(D,E)_{(\wp^0,\wp')}$
$= \{a,b,e,d,f\} \bigcup \{c\}$
$= \{a,b,c,d,e,f\}$.

5.2.4.2 Structure de $HSky$

Maintenant, voyons comment construire une structure $HSky$ (Figure. 5.5) pour stocker efficacement toutes les informations pré-calculées. Cette structure est isomorphe à celle de la Figure.5.4. Les préférences associées aux noeuds sont celles de la Figure.5.4. Notre objectif est d'éviter la construction d'une structure de données contenant tous les points skyline associés à chaque préférence hiérarchique possible sur l'espace de dimensions D. Nous proposons de stocker dans la structure $HSky$ les ensembles $HNSky$. Pour chaque arc, connectant un noeud fils associé à la préférence $\wp'$ à un noeud parent associé à la préférence $\wp$, nous calculons et stockons l'ensemble $HNSky(D,E)_{(\wp,\wp')}$.

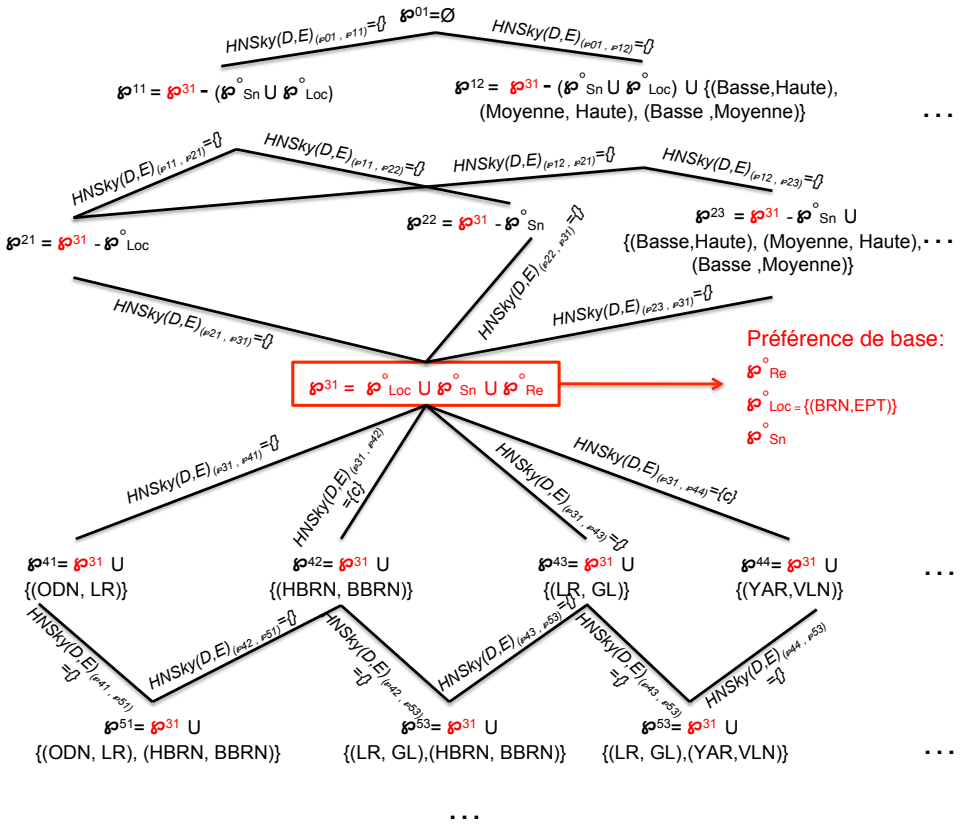


FIGURE 5.5 – La structure *HSky*

Algorithme 7 : $domine(\wp, x, y)$

input : x, y : deux points de l'ensemble E : ensemble de données, D : l'espace de dimensions, $\wp$: une préférence associée à D
output : dom : $bool$

```
1  dom ← faux
2  pour chaque ℘_{d_i} ∈ ℘ faire
3      si x <_{d_i} y
4      alors
5          dom ← vrai
6      sinon
7          dom ← faux
8          Sortir
9  Retourner dom
```

Algorithme 8 : $HNewSkySpe(\wp, \wp', Sky(D,E)_{\wp})$

input : E : ensemble de données, D : l'espace de dimensions, $\wp$, $\wp'$: deux préférences distinctes associées à D, $Sky(D,E)_{\wp}$: skyline associé à la préférence $\wp$
output : $HNSky$

```
1  HNSky ← ∅
2  Sky(D,E)_{℘'} ← Sky(D,E)_{℘}
3  pour chaque p ∈ Sky(D,E)_{℘} faire
4      pour chaque q ∈ Sky(D,E)_{℘} faire
           // Tester si q domine p avec la préférence ℘'
5          si domine(℘', q, p);                    // Algorithme 7
6          alors
7              Sky(D,E)_{℘'} ← Sky(D,E)_{℘'} − {p}
8              HNSky ← HNSky ⋃ {p}
9              Sortir
10 Retourner HNSky
```

L'algorithme 8 (resp. 9) décrit la méthode pour calculer cet ensemble lors de la navigation par spécialisation (resp. généralisation).
Tout d'abord, nous choisissons la préférence de base $\wp^0$ à partir de laquelle nous allons naviguer dans ses spécialisations ou ses généralisations. Ensuite, la construction de la structure $HSky$ se déroule en deux étapes (Algorithme 10) :

– Génération de toutes les spécialisations et généralisations de $\wp^0$ (Algorithme 10, lignes 2 et 3) et construction de la hiérarchie des préférences i.e. structure de spécialisations/généralisations (voir Figure. 5.4) qui donne la structure de $HSky$.

Algorithme 9 : $HNewSkyGen(\wp, \wp', Sky(D,E)_{\wp'})$

input : E : ensemble de données, D : l'espace de dimensions, $\wp$, $\wp'$: deux préférences distinctes associées à D, $Sky(D,E)_{\wp'}$: skyline associé à la préférence $\wp'$
output : $HNSky$

```
1  HNSky ← ∅
2  Sky(D,E)_℘ ← Sky(D,E)_℘'
3  pour chaque p ∈ E − Sky(D,E)_℘' faire
4      pour chaque q ∈ E − Sky(D,E)_℘' faire
           // Tester si p domine q avec la préférence ℘
5          si domine(℘, q, p);                          // Algorithme 7
6          alors
7              Sky(D,E)_℘ ← Sky(D,E)_℘ ⋃ {p}
8              HNSky ← HNSky ⋃ {p}
9              Sortir
10 Retourner HNSky
```

Le noeud au sommet de la structure de spécialisations/généralisations dans la Figure.5.4 correspond à la préférence $\varnothing$ sur les dimensions hiérarchiques,

– Pour chaque arc de $HSky$, calcul de l'ensemble $HNSky$ (Algorithme 10, lignes 5 et 6) enregistrant les points disqualifiés (resp. réintroduits) en passant d'une préférence à une spécialisation directe de cette préférence (resp. d'une préférence à une généralisation directe de cette préférence).

Les ensembles $HNSky$ sont générés de manière *Bottom-Up* pour les généralisations de $\wp^0$ (Algorithme 11), et de manière *Top-Down* pour les spécialisations de $\wp^0$ (Algorithme 12). En effet, grâce à la propriété 1, nous avons montré que lors d'une spécialisation ou d'une généralisation d'une préférence, certains des points skyline associés à la nouvelle préférence peuvent être dérivés sans test de dominance. Nous pouvons donc déduire, que chaque point de l'ensemble $Sky(D,E)_{\wp^0}$ associé au noeud $\wp^0$, appartient à l'ensemble skyline associé à l'un de ses noeuds parents (généralisation de $\wp^0$). Par conséquent, pour calculer l'ensemble $HNSky$ d'un noeud parent (généralisation de $\wp^0$) il n'est ni nécessaire de tester les points appartenant à l'ensemble $Sky(D,E)_{\wp^0}$, ni ceux appartenant aux ensembles $HNSky$ associés aux noeuds fils.

Pour calculer l'ensemble $HNSky$ associé à toute spécialisation de la préférence $\wp^0$, nous devons seulement tester les points appartenant à $Sky(D,E)_{\wp^0}$. En effet, grâce à la propriété 1, nous savons que le skyline associé à une spécialisation de $\wp^0$ est soit égal soit inclus dans $Sky(D,E)_{\wp^0}$. De plus, pour calculer l'ensemble $HNSky$ d'un noeud donné, il n'est pas nécessaire de tester les points supplémentaires appartenant aux ensembles $HNSky$ associés aux noeuds parents.

La structure $HSky$ obtenue est présentée dans la Figure.5.5. En pratique, la génération des spécialisations / généralisations et le calcul des $HNSky$ associés sont effectués

Algorithme 10 : Construction de la structure $HSky$

input : E : ensemble de données, D : espace de dimensions, $\wp^0$: préférence de base définie sur D, $Sky(D,E)_{\wp^0}$: skyline associé à $\wp^0$
output : structure $HSky$

$HSky \leftarrow \varnothing$
Générer toutes les spécialisations de $\wp^0$
Générer toutes les généralisations de $\wp^0$
Stocker toutes ces préférences dans la structure $HSky$
`// Générer tous les ensembles` $HNSky$ `des généralisations de` $\wp^0$
$HSky \leftarrow$ **BottomUpGeneration**($HSky$, $\wp^0$); `// Algorithme 11`
`// Générer tous les ensembles` $HNSky$ `des spécialisations de` $\wp^0$
$HSky \leftarrow$ **TopDownGeneration**($HSky$, $\wp^0$); `// Algorithme 12`
Retourner HSky

Algorithme 11 : BottomUpGeneration(HSky, $\wp^0$)

input : E : ensemble de données, D : espace de dimensions, $\wp^0$: préférence de base définie sur D, $Sky(D,E)_{\wp^0}$: skyline associé à $\wp^0$
output : $HSky$

ASC = ascendants($HSky$, $\wp^0$)`// Ensemble des ascendants directs de` $\wp^0$
pour chaque $\wp \in ASC$ **faire**
  `// Tester si` $\wp$ `est le label du noeud racine`
  si $\wp \neq$ *label(racine)* **alors**
    $HNSky(D,E)_{(\wp,\wp^0)}$ = **HNewSkyGen** ($\wp^0$, $\wp$, $Sky(D,E)_{\wp^0}$);
    `// Algorithme 9`
    Stocker $HNSky(D,E)_{(\wp,\wp^0)}$ dans la structure $HSky$
    $HSky \leftarrow$ **BottomUpGeneration**(HSky, $\wp$)
  Stocker $HNSky(D,E)_{(\wp,\wp^0)}$ dans la structure $HSky$
Retourner HSky

simultanément (un seul parcours de la hiérarchie liée à $\wp^0$).

5.2.4.3 Évaluation de requêtes

Dans l'analyse en ligne traditionnelle, les opérations *drill-down* et *roll-up* sont appliquées sur les hiérarchies des dimensions. Dans notre approche, ces opérations sont appliquées sur des préférences hiérarchiques associées à des requêtes skyline. Le théorème 3 nous permet d'assurer un calcul en ligne des points skyline, en évitant les tests de dominance, tout en réduisant ou en augmentant le niveau de détail des préférences associées.

Dans la suite, nous utilisons la structure $HSky$ décrite dans la Figure.5.6 pour illus-

Algorithme 12 : TopDownGeneration(HSky, $\wp^0$)

input : E : ensemble de données, D : espace de dimensions, $\wp^0$: préférence de base définie sur D, $Sky(D,E)_{\wp^0}$: skyline associé à $\wp^0$
output : $HSky$

```
1 DSC = descendants(HSky, ℘^0)// Ensemble des descendants directs de ℘^0
2 pour chaque ℘' ∈ DSC faire
      // Tester si ℘' est le label d'un noeud feuille
3     si IsNotLeaf(℘') alors
4         HNSky(D,E)_(℘^0,℘') = HNewSkySpe (℘^0, ℘', Sky(D,E)_℘^0);
          // Algorithme 8
5         Stocker HNSky(D,E)_(℘^0,℘') dans la structure HSky
6         HSky ← TopDownGeneration(HSky, ℘')
7     Stocker HNSky(D,E)_(℘^0,℘') dans la structure HSky
8 Retourner HSky
```

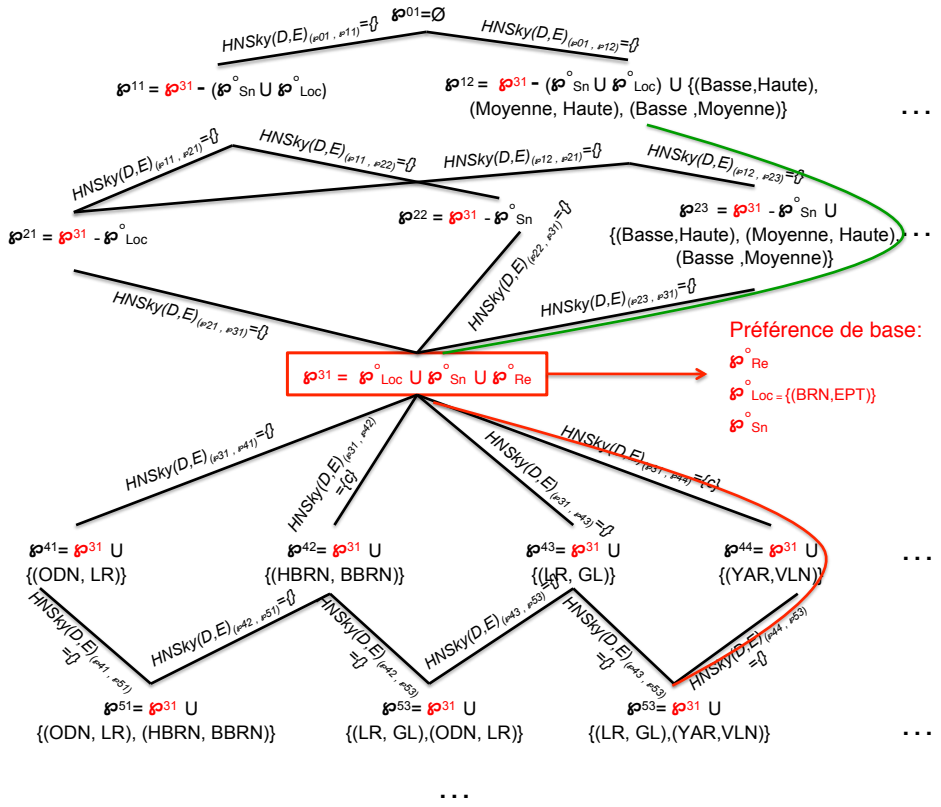


FIGURE 5.6 – Illustration du théorème 3

trer la navigation à partir du skyline de la préférence de base $\wp^0 = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\} \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$. Le skyline relatif à $\wp^0$ est $Sky(D,E)_{\wp^0} = \{a, b, e, c, d, f\}$.

Nous détaillons comment calculer, en utilisant le théorème 3, les points skyline associés aux préférences $\wp' = \wp^0 \bigcup \{(LR,GL),(YAR,VLN)\}$ (une spécialisation de $\wp^0$), et $\wp = \{(Moyenne,Haute),(Basse,Moyenne), (Basse,Haute)\} \bigcup \wp^0_{Re}$ (une généralisation de $\wp^0$). L'exploration de la structure $HSky$ est guidée par une heuristique (Algorithme 13, lignes 5 et 15).

Skyline d'une préférence spécialisée Le skyline associé à la préférence $\wp'$, avec $\wp'$ une spécialisation de la préférence de base $\wp^0$, est calculé comme suit.
La navigation commence à partir du noeud associé à la préférence de base $\wp^0$ et explore les noeuds fils, à la recherche du noeud associé à la préférence $\wp'$ selon l'algorithme 13 (ligne 17). Si le noeud objectif est un descendant direct de $\wp^0$, la recherche s'arrête. Le skyline de la préférence $\wp'$ est alors simplement calculé par la soustraction de l'ensemble $HNSky(D,E)_{(\wp^0,\wp')}$ de $Sky(D,E)_{\wp^0}$ (Algorithme 13, lignes de 18 à 20). Dans le cas contraire, dès que la recherche atteint un noeud ayant pour préférence une généralisation de $\wp'$, il suffit de poursuivre la recherche sur le sous-graphe courant. Dans ce cas de figure, le skyline associé à la préférence $\wp'$ correspond à la soustraction des ensembles $HNSky$ de chaque noeud visité de l'ensemble $Sky(D,E)_{\wp^0}$ (Algorithme 13, lignes de 22 à 24).

Exemple 34 $\wp^0 = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\} \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$.
Le skyline relatif à $\wp^0$ *est* $Sky(D,E)_{\wp^0} = \{a, b, e, c, d, f\}$.
Soit $\wp' = \wp^0 \bigcup \{(LR,GL), (YAR,VLN)\}$ *une spécialisation de* $\wp^0$.
Dans la Figure.5.6, $\wp^0$ *correspond à* $\wp^{31}$ *et* $\wp'$ *correspond à* $\wp^{53}$. *Le chemin parcouru pour calculer le skyline associé à* $\wp'$ *est indiqué en rouge.*
$Sky(D,E)_{\wp'} = Sky(D,E)_{\wp^{31}}$ - $(HNSky(D,E)_{(\wp^{31},\wp^{44})} \bigcup HNSky(D,E)_{(\wp^{44},\wp^{53})})$
$= \{a, b, c, e, d, f\}$ -$(\{c\} \bigcup \{\})$
$= \{a, b, e, d, f\}$.

Skyline d'une préférence généralisée Le skyline associé à la préférence $\wp$, avec $\wp$ une généralisation de la préférence de base $\wp^0$, est calculé comme suit.
La navigation commence à partir du noeud associé à la préférence de base $\wp^0$ et explore les noeuds parents, à la recherche du noeud associé à la préférence $\wp$ selon l'algorithme 13 (ligne 7). Si le noeud objectif est un ascendant direct de $\wp^0$, la recherche s'arrête. Le skyline de la préférence $\wp$ est alors simplement calculé par l'union de l'ensemble $Sky(D,E)_{\wp^0}$ et l'ensemble $HNSky(D,E)_{(\wp,\wp^0)}$ (Algorithme 13, lignes de 8 à 10). Dans le cas contraire, dès que la recherche atteint un noeud ayant pour préférence une spécialisation de $\wp$, il suffit de poursuivre la recherche sur le sous-graphe courant. Dans ce cas de figure, le skyline associé à la préférence $\wp$ correspond à l'union des ensembles $HNSky$ de chaque noeud visité, union l'ensemble $Sky(D,E)_{\wp^0}$ (Algorithme 13, lignes de 12 à 14).

Exemple 35 $\wp^0 = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (YAR, EPT), (OD, EPT), (FRM, EPT), (ILL, EPT), (VLN, EPT), (COS, EPT)\} \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$.
Le skyline relatif à $\wp^0$ *est* $Sky(D, E)_{\wp^0} = \{a, b, e, c, d, f\}$.
Soit $\wp = \{(Moyenne, Haute), (Basse, Moyenne), (Basse, Haute)\} \bigcup \wp^0_{Re}$ *une généralisation de* $\wp^0$.
Dans la Figure.5.6, $\wp^0$ *correspond à* $\wp^{31}$ *et* $\wp$ *correspond à* $\wp^{12}$. *Le chemin parcouru pour calculer le skyline associé à* $\wp$ *est indiqué en vert.*
$Sky(D, E)_{\wp} = Sky(D, E)_{\wp^{31}} \bigcup HNSky(D, E)_{(\wp^{23}, \wp^{31})} \bigcup HNSky(D, E)_{(\wp^{12}, \wp^{23})}$
$= \{a, b, c, e, d, f\} \bigcup \{\} \bigcup \{\}$
$= \{a, b, c, d, e, f\}$.

5.2.4.4 Évaluations expérimentales

Dans cette section, nous présentons une évaluation expérimentale de notre algorithme *HSky* sur des données synthétiques. L'algorithme *HSky* a été implémenté en langage *JAVA*. Nous avons conduit nos expérimentations sur un Intel Xeon CPU 3GHz et 16 GB de RAM sous une plateforme Linux. Pour les dimensions à un seul niveau hiérarchique, les données ont été produites par le générateur publié par les auteurs de [BKS01]. Trois types de jeux de données ont été générés : des données indépendantes, des données corrélées et des données non-corrélées. La description de ces jeux de données peut être trouvée dans [BKS01]. Comme dans le chapitre 4, nous présentons seulement les résultats relatifs aux données non-corrélées. Les résultats relatifs aux données indépendantes et aux données corrélées sont similaires, mais les temps de pré-calcul et de réponse aux requêtes sont beaucoup plus courts pour les données corrélées. Les dimensions hiérarchiques ont été générées selon une distribution de Zipfian [Tre94]. Les structures hiérarchiques des différentes dimensions étaient décrites dans un fichier XML. Par défaut, nous avons initialisé le paramètre Zipfian θ à 1 (données non-corrélées). Nous avons obtenus 700.000 tuples avec 6 dimensions à un seul niveau hiérarchique. Nous avons fait varier le nombre de dimensions hiérarchiques de 3 à 20 et le nombre de niveaux hiérarchiques de ces dimensions de 3 à 7. Nous avons aussi fait varier le nombre de successeurs des différentes valeurs des dimensions hiérarchiques (sauf pour les valeurs feuilles) de 3 à 10. Nous avons choisi une préférence de base de sorte que le nombre de spécialisations soit équilibré avec le nombre de généralisations. Le template de la préférence posée dans la requête a été choisi de sorte que cette préférence représente une spécialisation / généralisation indirecte de la préférence de base.

Comme il n'existe pas, à notre connaissance, de travaux dans la littérature conciliant extraction de points skyline et prise en compte de dimensions hiérarchiques. Nous avons implémenté l'algorithme *DC-H*, basée sur le principe de l'algorithme de calcul de skyline *Divide & Conquer* présenté précédemment dans le chapitre 2. *DC-H* ne matérialise aucun résultat partiel, il recalcule à chaque niveau hiérarchique l'ensemble des skyline.

Étant donné que *Divide & Conquer* est une façon naturelle d'exprimer des algorithmes parallèles, et afin d'améliorer les performances de *DC-H*, nous avons décidé de paralléliser le calcul de *DC-H*. L'algorithme *HSky* est basé sur le principe de *DC-H* pour le calcul des skyline.

Algorithme 13 : $HSky(\wp^0, \wp, Sky(D,E)_{\wp^0})$

input : Le noeud courant, $\wp^0$: la préférence de base associée au noeud courant, $Sky(D,E)_{\wp^0}$, $\wp$: la nouvelle préférence, structure $HSky$, $Sky(D,E)$: le skyline associé à la préférence $\varnothing$
output : $Sky(D,E)_{\wp}$

```
1  Sky ← ∅
2  Sky ← Sky(D,E)_{℘^0}
3  si ℘ = ∅ alors
4  |  Sky(D,E)_{℘} ← Sky(D,E)
5  sinon
   |  // Tester si ℘ est une généralisation de ℘^0
6  |  si ℘ ⊂_h ℘^0 alors
7  |  |  pour chaque noeud parent ℘^1 de ℘^0 faire
8  |  |  |  si ℘ = ℘^1 alors
9  |  |  |  |  Sky ← Sky(D,E)_{℘^0} ⋃ HNSky(D,E)_{(℘^1,℘^0)}
10 |  |  |  |  Sky(D,E)_{℘} ← Sky
11 |  |  |  sinon
12 |  |  |  |  si ℘ ⊂_h ℘^1 alors
13 |  |  |  |  |  Sky ← Sky(D,E)_{℘^0} ⋃ HNSky(D,E)_{(℘^1,℘^0)}
14 |  |  |  |  |  HSky(℘^1, ℘, Sky)
15 |  |  |  |  |  Sortir
   |  // Tester si ℘ est une spécialisation de ℘^0
16 |  si ℘^0 ⊂_h ℘ alors
17 |  |  pour chaque noeud fils ℘^1 de ℘^0 faire
18 |  |  |  si ℘ = ℘^1 alors
19 |  |  |  |  Sky ← Sky(D,E)_{℘^0} - HNSky(D,E)_{(℘^0,℘^1)}
20 |  |  |  |  Sky(D,E)_{℘} ← Sky
21 |  |  |  sinon
22 |  |  |  |  si ℘^1 ⊂_h ℘ alors
23 |  |  |  |  |  Sky ← Sky(D,E)_{℘^0} - HNSky(D,E)_{(℘^0,℘^1)}
24 |  |  |  |  |  HSky(℘^1, ℘, Sky)
25 |  |  |  |  |  Sortir
26 Retourner Sky(D,E)_{℘}
```

Dans ces expérimentations, nous comparons les performances de l'algorithme $HSky$ avec l'algorithme DC-H.

Paramètre	Valeur
Nb. de tuples	100K
Nb. de dimensions à un seul niveau hiérarchique	6
Nb. de dimensions hiérarchiques	6
Nb. de successeurs de chaque valeur d'une dimension hiérarchique	3
Nb. de niveaux d'une dimension hiérarchique	3
Paramètre Zipfian θ	1

TABLE 5.2 – Valeurs par défaut

Variation du nombre de dimensions hiérarchiques Dans cette première expérimentation, le nombre de dimensions à un seul niveau hiérarchique est fixé à 6 et le nombre de dimensions hiérarchiques varie de 3 à 20. La Figure.5.7 montre que la taille mémoire et le temps de pré-calcul de $HSky$ augmentent avec le nombre de dimensions hiérarchiques. Cela est dû à la complexité de la structure de spécialisations / généralisations construite par $HSky$ qui induit certes une augmentation substantielle du taux de stockage mais qui évolue lentement (i.e. évolution quadratique). L'algorithme *DC-H* ne nécessite aucun stockage mémoire, ni aucun temps de pré-calcul étant donné qu'il ne matérialise aucun résultat partiel. Cependant, $HSky$ surpasse *DC-H* en terme de temps de réponse aux requêtes (Figure.5.7c). En effet, $HSky$ affiche des temps de réponse quasi-instantanés alors que pour *DC-H* ils augmentent sensiblement avec le nombre de dimensions hiérarchiques. Cela s'explique par le fait que *DC-H* recalcule à chaque nouvelle requête l'ensemble des skyline, alors qu'il est déduit à partir de simples opérations de soustraction et d'union dans $HSky$. En effet, les tests de dominance augmentent avec le nombre de dimensions hiérarchiques. Le temps de réponse aux requêtes est un critère d'évaluation très important car nous nous positionnons dans un contexte d'entrepôts de données et d'analyse en ligne. $HSky$ est donc plus performant que *DC-H*.

Variation de la taille de l'ensemble de données

Dans cette expérimentation, le nombre de tuples de l'ensemble de données varie de 50.000 à 700.000. La Figure.5.8 montre que la taille mémoire et le temps de pré-calcul de $HSky$ augmentent avec la taille de l'ensemble de données. Cela s'explique par le fait que la taille des informations stockées et analysées croit avec l'augmentation du volume des données (i.e. les points skyline stockés dans les ensembles $HNSky$), ce qui induit une évolution polynomiale (i.e. lente) du taux de stockage. Cependant, notre méthode est plus performante que *DC-H* en terme de temps de réponse aux requêtes.

Variation du nombre de niveaux hiérarchiques des dimensions hiérarchiques

Nous avons fait varier le nombre de niveaux hiérarchiques des dimensions hiérarchiques de 3 à 7. La Figure.5.9 montre que la taille mémoire et le temps de pré-calcul de $HSky$ augmentent avec le nombre de niveaux hiérarchiques. Cette augmentation peut

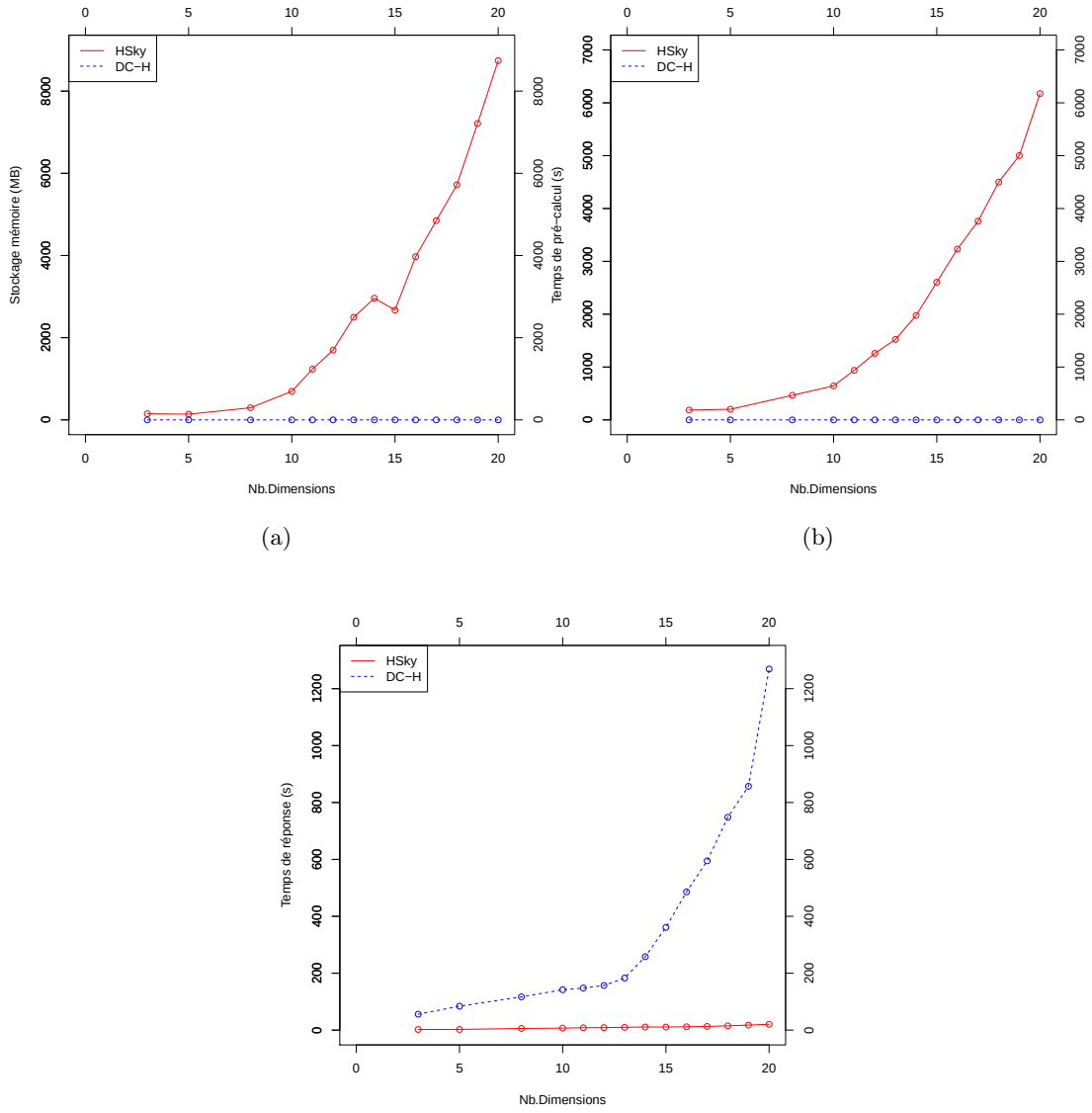


FIGURE 5.7 – Variation du nombre de dimensions hiérarchiques

s'expliquer par l'évolution exponentielle du nombre de valeurs des dimensions hiérarchiques induite par l'augmentation du nombre de niveaux hiérarchiques associé à ces dernières. Nous pouvons cependant noter que ces expérimentations ont été poussées assez loin étant donné que dans les applications réelles il est rare de trouver des dimensions avec 7 niveaux hiérarchiques. Une fois de plus, notre méthode est plus performante que *DC-H* en terme de temps de réponse aux requêtes.

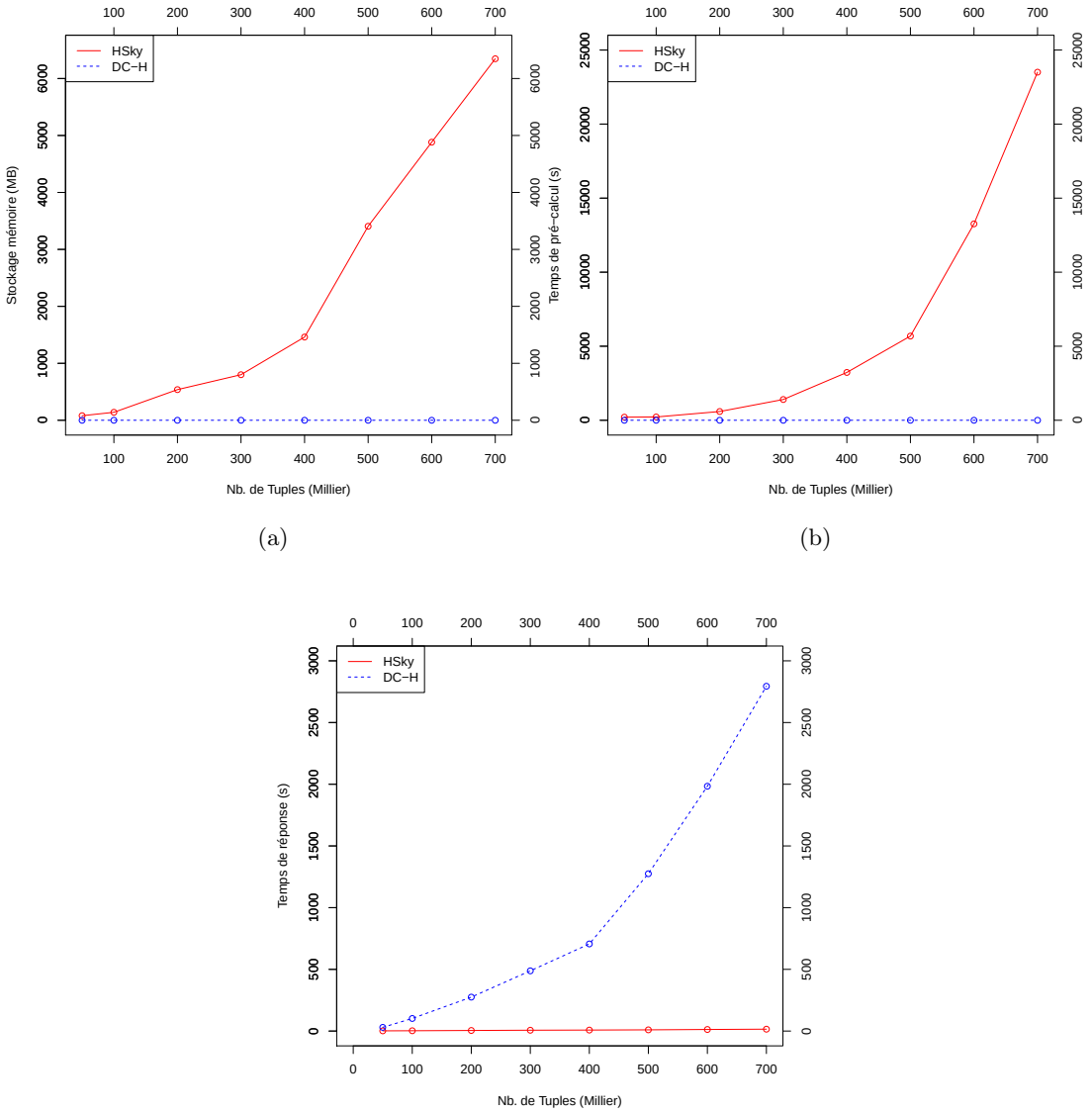


FIGURE 5.8 – Variation de la taille de l'ensemble de données

Variation du nombre des successeurs des valeurs des dimensions hiérachiques

Nous avons fait varier le nombre des successeurs (i.e. valeurs spécialisées) des valeurs des dimensions hiérarchiques de 3 à 10. La Figure.5.10 montre que la taille mémoire et le temps de pré-calcul de *HSky* augmentent lentement avec le nombre de niveaux hiérarchiques. Cette augmentation peut s'expliquer par l'évolution polynomiale du nombre de valeurs des dimensions qui implique une évolution polynomiale lente du nombre de

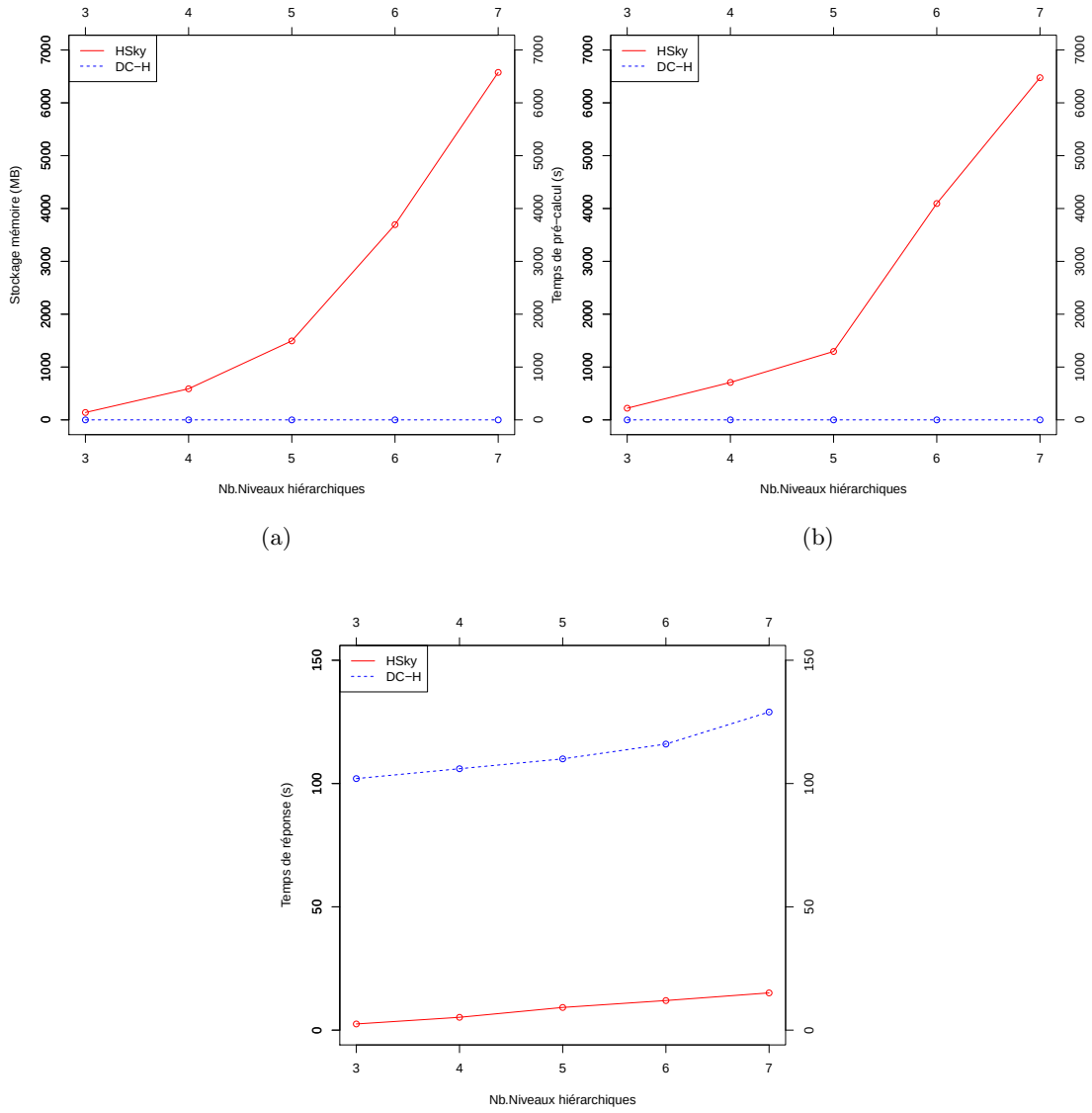


FIGURE 5.9 – Variation du nombre de niveaux d'une dimension hiérrachique

noeuds dans la structure de spécialisations / généralisations. Encore une fois, notre méthode est plus performante que *DC-H* en terme de temps de réponse aux requêtes.

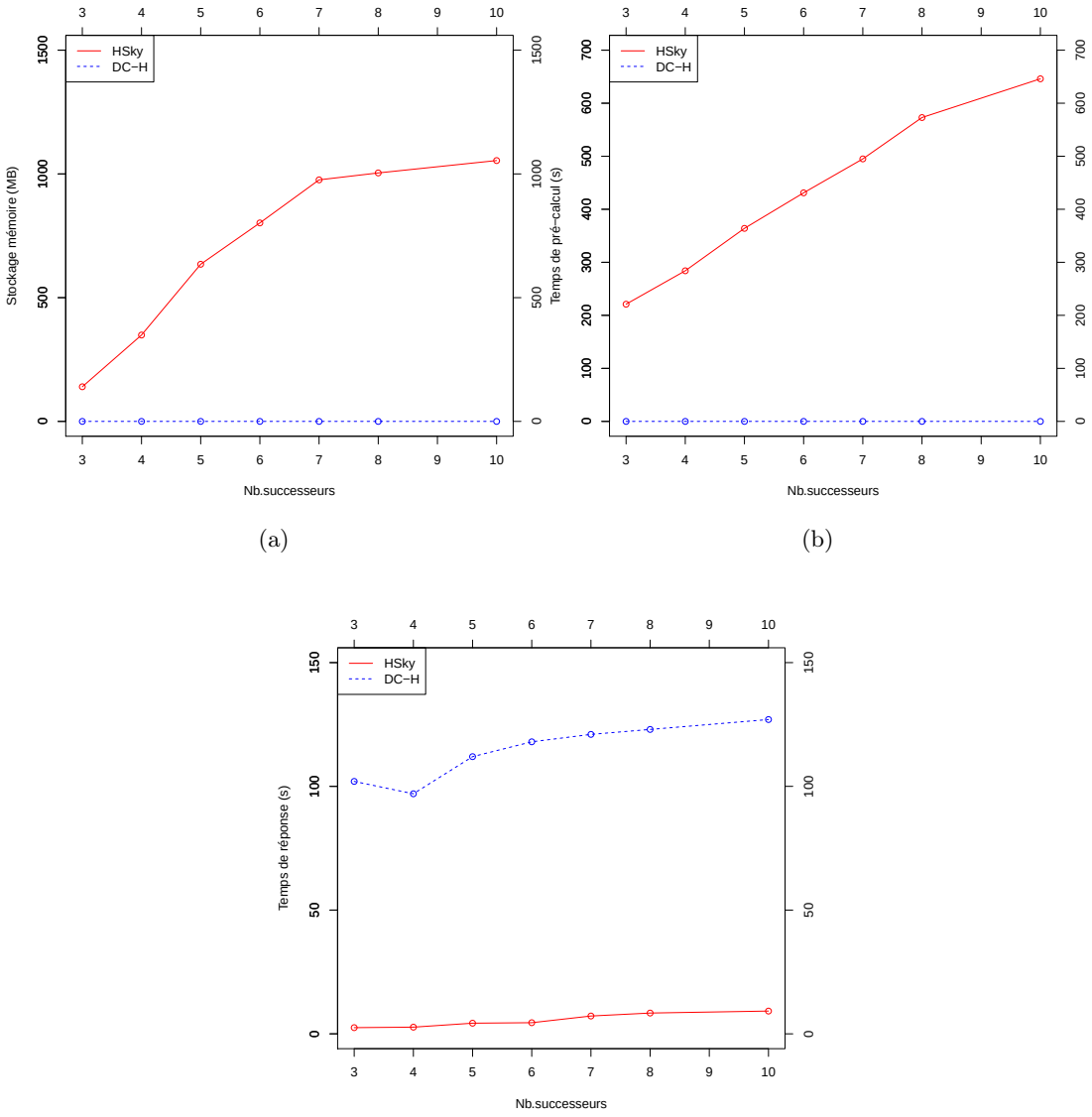


FIGURE 5.10 – Variation du nombre des successeurs des valeurs d'une dimension hiérarchique

5.3 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode *HSky* qui étend la recherche des points skyline aux dimensions hiérarchiques. La définition de hiérarchies sur les dimensions permet à l'utilisateur d'affiner ou de généraliser ses préférences lors du calcul des skyline. Nous avons mis en évidence un ensemble de propriétés permettant la

formalisation des relations hiérarchiques entre les préférences associées aux différentes dimensions. Ces propriétés permettent notamment aux utilisateurs de naviguer le long des relations hiérarchiques entre les préférences tout en assurant un calcul en ligne des points skyline correspondants. En particulier, la propriété de monotonie hiérarchique a contribué considérablement à l'analyse et au calcul des points skyline en présence de préférences associées à des dimensions hiérarchiques. En effet, cette propriété indique que tout point skyline associé à une préférence donnée reste skyline lorsque nous considérons une généralisation de cette préférence. Cette propriété nous a permis de caractériser les points skyline qui restent skyline et ceux qui ne le sont plus, lors de la spécialisation (drill-down) ou de la généralisation (roll-up) des préférences associées aux dimensions hiérarchiques. Ceci nous a permis de réduire le nombre de tests de dominance pour calculer efficacement les skyline lors de la navigation sur les dimensions hiérarchiques.

Cette méthode, $HSky$, respecte parfaitement tous les critères d'évaluation d'un algorithme de calcul de skyline (i.e. Efficacité, Progressivité, Préférences, Complétude et Correction) décrits précédemment dans la sous-section 2.2.2 du chapitre 2.

Les expérimentations réalisées dans cette étude soulignent l'efficacité des performances de $HSky$ comparé à la méthode DC-H, même si la construction de la structure de $HSky$ implique un coût en terme de temps de pré-calcul et de stockage mémoire. Pour que l'utilisation de cette structure soit bénéficiaire, il faut un certain nombre d'interactions utilisateur (6 à 8 requêtes en moyenne). Cependant, cette structure a vocation à évoluer selon le contexte d'application. Par exemple, si la taille de la structure est trop importante, il est possible de restreindre les spécialisations / généralisations aux valeurs explicitement ordonnées dans la préférence de base. Nous souhaitons étudier le cas des dimensions hiérarchiques associées à des préférences dynamiques (cf. chapitre 4), dans ce cas de figure l'aspect dynamique des préférences peut occasionner une explosions du nombre de préférences à matérialiser.Il est alors possible de restreindre les préférences spécialisables / généralisables aux préférences d'ordre n étudiées dans le chapitre 4.

Afin de démontrer l'utilité de notre méthode, nous avons appliqué notre algorithme sur des jeux de données réels, notamment dans le cadre de l'analyse multidimensionnelle des résultats de simulations stockés dans l'entrepôt de données N-$Catch$ décrit dans le chapitre 3. Ce travail fera l'objet de la deuxième partie du prochain chapitre 6.

Quatrième partie

Couplage de l'entrepôt *N-Catch* avec les requêtes skyline

Chapitre 6

Utilisation des algorithmes EC^2Sky et $HSky$ à partir de N-$Catch$

.

6.1 Introduction et motivations

Les opérateurs OLAP permettent de naviguer dans l'entrepôt de données *N-Catch* dans le but de trouver des informations intéressantes et d'exploiter au mieux les données de simulations qui y sont archivées. La difficulté pour l'utilisateur est qu'il doit trouver par lui même ces informations ainsi que les régions intéressantes de l'entrepôt. Or, la densité et le gros volume de *N-Catch* empêche une exploration intuitive des données.

L'utilisateur a donc besoin de nouveaux outils pour le guider vers les régions intéressantes de l'entrepôt, en prenant en compte ses préférences. Confronté à un important volume d'information, les techniques de recherche d'information et l'identification de domaines de préférences nous semblent être de bonnes pistes pour répondre à ces attentes.

Nous nous sommes intéressés particulièrement à une méthode de recherche d'information à savoir *les requêtes skyline* (cf. chapitre 2) qui nous parait l'approche la plus adaptée à notre problématique. Pour la sélection de cette méthode nous nous sommes basés sur les deux critères suivants :

- Prendre en compte ou pouvoir s'adapter à des données multidimensionnelles et hiérarchiques,
- Prendre en compte les préférences utilisateurs de manière explicite.

Pour rappel, les requêtes skylines sont des requêtes multicritères (i.e. multidimensionnelles) avec préférences. Elles permettent de sélectionner les réponses les meilleures au vu des préférences. Cela correspond aux préoccupations des utilisateurs de l'entrepôt de données *N-Catch*. Nous avons donc développé deux méthodes basées sur les requêtes skyline :

- La première méthode EC^2Sky (cf. chapitre 4) permet un calcul incrémental des skyline en présence de préférences dynamiques. Les préférences dynamiques sont des préférences qui changent d'un utilisateur à un autre. Par exemple, un utilisateur préfére analyser les parcelles les plus polluées avec *Prairie*, tandis qu'un autre utilisateur préfére analyser les parcelles avec *CIPAN*,

- La deuxième méthode $HSky$ (cf. chapitre 5) permet de coupler les opérateurs OLAP "drill-down" et "roll-up" avec les requêtes skyline, afin que l'utilisateur puisse naviguer le long des dimensions hiérarchiques (i.e. spécialiser / généraliser), en assurant un calcul en ligne des skyline. Ainsi, $HSky$ permet de naviguer dans la hiérarchie associée à la dimension agricole de l'entrepôt de données N-$Catch$. Par exemple, soit la requête *pour les itinéraires techniques (ITK) de la culture Blé, quels sont ceux pour lesquels les émissions de l'azote vers l'eau sont les plus fortes / les plus faibles ?* Suivant les résultats obtenus, nous pouvons être intéressés par affiner un peu plus l'analyse des ITK de *Blé* remarquables au niveau opération. Par exemple, si un ITK donné est très polluant, il serait intéressant de regarder au niveau de l'opération culturale si ce constat est vérifié pour toutes les opérations de cet ITK ou seulement pour certaines d'entre elles. De même, si un ITK est faiblement polluant, nous pouvons regarder si à un niveau plus général (niveau culture par exemple) cela se vérifie aussi. Autrement dit, un ITK faiblement polluant implique-t-il que la culture à laquelle il appartient soit faiblement polluante par rapport aux autres cultures ? Toutes ces navigations sur la dimension agricole peuvent être accomplies de manière très rapide grâce à la méthode $HSky$.

Dans ce chapitre, nous présentons le couplage de l'entrepôt de données N-$Catch$ avec les algorithmes EC^2Sky et $HSky$. Afin de démontrer l'utilité de nos méthodes, nous appliquons ces algorithmes sur les données de simulations issues de TNT, dans le cadre de l'analyse multidimensionnelle des données stockées dans l'entrepôt de données N-$Catch$.

6.2 Couplage de N-$Catch$ avec les algorithmes EC^2Sky et $HSky$

Dans cette section, nous montrons l'intérêt d'associer des préférences dynamiques et hiérarchiques à l'analyse skyline dans le contexte de l'application N-$Catch$. Pour ce faire, un ensemble de questions thématiques, auxquelles les méthodes EC^2Sky et $HSky$ couplées avec l'entrepôt de données N-$Catch$ peuvent apporter des réponses, a été constitué.

6.2.1 Requêtes étudiées

Un échange avec les utilisateurs de *N-Catch*, et en particulier les chercheurs en agro-hydrologie de l'UMR SAS/INRA Rennes, a permis de rassembler un ensemble de questions auxquelles le couplage de l'entrepôt de données *N-Catch* avec les algorithmes EC^2Sky et $HSky$ pouvait apporter des réponses. Ces questions portent sur des interactions spatio-temporelles, des interactions milieu-pratiques agricoles. Les questions qui ont parues possibles et intéressantes à traiter sont listées ci-dessous :

1. Quelles sont les parcelles les moins / plus polluées en préférant les *Prairie permanentes* aux *Prairie en rotation* ?
2. Quelles sont les parcelles les plus polluées en préférant les *Prairie avec fauche* aux prairie *sans fauche* ?
3. Quelles sont les parcelles les plus polluées en préférant celles avec *CIPAN (Cultures Intermédiaires Pièges À Nitrate)* ?
4. Quelles sont les parcelles les plus polluées parmi celles qui sont, les plus proches du ruisseau et avec *Prairie* ?
5. Pour les itinéraires techniques de *Blé*, quels sont ceux qui polluent le plus / le moins ?
6. Pour les itinéraires techniques de *Blé* dont les émissions de l'azote vers l'eau sont les plus fortes, quand interviennent ces pertes ?

Dans ce qui suit, nous illustrons le couplage de l'entrepôt de données *N-Catch* avec les algorithmes EC^2Sky et $HSky$ sur les questions n^o1 et n^o5. Nous avons choisi de détailler le traitement de ces questions, car elles portent sur différents niveaux de granularité des trois dimensions de *N-Catch*. Ceci nous permet d'illustrer la navigation le long des axes hiérarchiques des trois dimensions. La variable qui a été choisie est $DNRU$ qui représente les variations de la quantité d'azote stockée dans la réserve utile du sol. Cette variable représente la variation de stock d'azote dans la réserve utile du sol. C'est un indicateur partiel de l'émission d'azote vers la nappe, puisque les variations de stock peuvent provenir soit de l'absorption par les plantes, soit de l'émission vers la nappe. Par simplification on dira qu'il caractérise des situations "plus" ou "moins pollués".

6.2.2 Illustration du couplage de *N-Catch* avec EC^2Sky

Dans cette sous-section, nous allons montrer comment traduire les questions formulées ci-dessous en requête skyline, et les préférences associées en préférence d'ordre n (cf. chapitre 4).

1. *Quelles sont les parcelles les moins polluées en préférant les Prairies permanentes (Prairie-p) aux Prairies en rotation (Prairie-r)* ?
   Cette question illustre des interactions milieu-pratiques agricoles. Pour la traiter nous commençons d'abord par dégager la préférence qui y est exprimée. Ensuite

nous appliquons l'algorithme EC^2Sky sur les données de simulation stockées dans N-$Catch$ afin d'extraire les points skyline associés à cette préférence. Dans ce cas de figure, la question porte sur deux dimensions (au sens skyline i.e. critères) : la dimension agricole (*Pratique Agricole*) et la mesure $DNRU$. La fonction d'agrégation utilisée pour le calcul de cette mesure est la moyenne des cumuls annuels à l'échelle de la parcelle. La préférence exprimée sur la mesure $DNRU$ est un ordre total $\leqslant$ qui spécifie qu'on préfère les parcelles ayant la valeur $DNRU$ la moins élevée. La préférence exprimée sur la dimension *Pratique Agricole (PA)* correspond à la préférence d'ordre 1 suivante : $Prairie\text{-}p <_{PA} *$. Cette préférence signifie que les parcelles avec *Prairies permanentes* sont préférées à toutes les autres, i.e. aux *Prairies en rotation*.

Pour cette préférence, l'algorithme EC^2Sky a retourné 19 parcelles qui appartiennent au skyline sur un total de 4620 parcelles. Ces résultats sont décrits dans la table 6.1, et visualisés dans *QGis* (Figure 6.1). Dans la table 6.1, pour des raisons de lisibilité nous avons décrit uniquement les 5 premières parcelles ayant les valeurs $DNRU$ les moins élevées. Il est intéressant de noter qu'une partie des parcelles retournées par la requête ont comme type de culture *Prairies en rotation* alors que l'utilisateur avait spécifié son intérêt pour les *Prairies permanentes*. Contrairement aux requêtes $OLAP$ qui n'auraient retourné que les parcelles avec *Prairies permanentes*, les requêtes skyline retournent toutes les parcelles susceptibles d'intéresser l'utilisateur. Elles retournent donc toutes les parcelles ayant des valeurs $DNRU$ moins élevées que la parcelle avec *Prairies permanentes*.

Les temps de calcul associés à cette requête sont de l'ordre de quelques millièmes de seconde pour le temps de pré-traitement (i.e. construction de la structure EC^2Sky), et pour la satisfaction de la requête. Lorsque nous raffinons la préférence associée à la dimension agricole, le temps de pré-traitement et de satisfaction de la requête sont de l'ordre de millièmes de seconde. Cela montre bien l'efficacité de notre méthode EC^2Sky dans le cadre de l'application N-$Catch$.

De la même manière, nous avons calculé la requête suivante :
Quelles sont les parcelles les plus polluées en préférant les Prairies permanentes (Prairie-p) aux Prairies en rotation (Prairie-r) ?
Les résultats relatifs à cette requête sont décrits dans la table 6.2 et la Figure.6.2. Pour cette préférence, l'algorithme EC^2Sky a retourné 2 parcelles appartenant au skyline sur un total de 4620 parcelles. L'ensemble des parcelles retourné par les requêtes skyline varie d'une requête à une autre. La cardinalité des ensembles skyline retournés jusqu'à présent est petite (entre 2 et 19 parcelles) car avons des dimensions à faible cardinalité (2 pour la dimension agricole), ce qui constitue un critère discriminant pour la disqualification de parcelles non skyline.

2. *Pour les itinéraires techniques de Blé, quels sont ceux qui polluent le moins ?*
La question porte sur deux critères : agricole (*Pratique Agricole*) et la mesure $DNRU$. La fonction d'agrégation utilisée pour le calcul de cette mesure est la moyenne annuelle des valeurs journalières à l'échelle de l'ITK et du bassin versant. La préférence exprimée sur la mesure $DNRU$ correspond à un ordre total $\leqslant$ qui

spécifie qu'on préfère les parcelles ayant la valeur $DNRU$ la moins élevée. La préférence exprimée sur la dimension *Pratique Agricole* correspond à la préférence d'ordre 1 suivante : $Ble <_{PA} *$. Cette préférence signifie que les ITK de la culture *Blé* sont préférés aux ITK de tous les autres types de cultures.

Pour cette préférence, l'algorithme EC^2Sky a retourné 2 ITK appartenant au skyline sur un total de 2642 ITK. Ces résultats sont décrits dans la table 6.3.

Les temps de calcul associés à cette requête sont de l'ordre de millièmes de seconde pour le temps de pré-traitement (i.e. construction de la structure EC^2Sky), et pour la satisfaction de la requête. De la même manière, nous avons calculé la requête suivante :
Pour les itinéraires techniques de Blé, quels sont ceux qui polluent le plus ?
Les résultats relatifs à cette requête sont décrits dans la table 6.4. Pour cette préférence, l'algorithme EC^2Sky a retourné 8 ITK appartenant au skyline sur un total de 2642 ITK. Nous constatons que le seul ITK de *Blé* extrait par la requête (i.e. le plus polluant) a une valeur $DNRU$ quasi nulle, et peut donc être considéré comme non polluant. Nous pouvons ainsi conclure que les ITK de *Blé* sont non polluants. Par ailleurs, la requête a extrait des ITK d'autres types de cultures qui polluent plus que les ITK de *Blé* et qui sont donc susceptibles d'intéresser l'utilisateur.

Nous raffinons la préférence associée à la dimension agricole en préférence d'ordre 2 comme suit : $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$. Les résultats associés à cette nouvelle préférence sont décrits dans la table 6.5. Pour cette préférence, l'algorithme EC^2Sky a retourné 4 ITK (les ITK 205, 287, 566 et 678 sont disqualifiés du skyline associé à la nouvelle préférence). Le temps de pré-calcul et le temps de réponse à la requête sont inférieurs à la seconde. Encore une fois, cela montre bien l'efficacité d'un point de vue du calcul de notre méthode EC^2Sky dans le cadre de l'application *N-Catch*.

6.2.3 Illustration du couplage de *N-Catch* avec *HSky*

Dans cette sous-section, nous allons montrer comment l'utilisateur peut spécialiser ou généraliser ses préférences tout en assurant un calcul efficace des skyline grâce à la méthode $HSky$ (cf. chapitre 5).

Dans la sous-section précédente, la requête de l'utilisateur était la suivante :
Pour les itinéraires techniques de Blé, quels sont ceux qui polluent le plus ? (cf. table 6.4)
Cette requête a extrait les ITK les plus polluants liés aux différentes cultures qui in-

1. Blé avec apport de type lisier de porc (LP)
2. Retournement de la prairie avec libération de matière organique
3. Apport fumier de volaille (FV) suivi d'un semis de maïs ensilage
4. Pomme de terre avec apport de type fumier de bovin (FB)
5. Semis de RGA associé à du trèfle blanc de type TBmoins
6. Semis de RGA associé à du trèfle blanc de type TBmoins en automne (aut)
6. Semis de RGA avec apport de type lisier de porc en automne

ID Parcelle	Type culture	DNRU (Kg/Ha/Année)	Rotation	Type-sol
406	prairie-r	0,03	(prairie-t-rga, colza, blé, orge, maïs, blé, cipan-rgi, maïs, cipan-rgi, pois)	4
554	prairie-r	0,03	(prairie-t-rga, blé, maïs, orge, maïs, orge, blé, orge, blé, maïs)	4
879	prairie-r	0,03	(prairie-t-rga, blé, moutarde, maïs, cipan-rgi, maïs, blé, orge)	3
4126	prairie-r	0,03	(prairie-t-rga, blé, orge, blé)	4
228	prairie-p	0,04	prairie-p-rga	4

TABLE 6.1 – Les 5 parcelles les moins polluées avec la préférence $Prairie - p <_{PA} *$

ID Parcelle	Type culture	DNRU (Kg/Ha/Année)	Rotation	Type-sol
3579	prairie-r	4,25	(prairie-t-rga, maïs, blé, cipan-rgi, maïs)	4
7	prairie-p	1,5	prairie-p-rga	3

TABLE 6.2 – Les parcelles les plus polluées avec la préférence $Prairie - p <_{PA} *$

ID ITK	Type ITK	Culture	DNRU (Kg/Ha/Année)	Durée (j)
107	blé-LP[1]	blé	-0,33	270
721	ret-aut-rga[2]	prairie-t-rga	-193,45	1

TABLE 6.3 – Les ITK les moins polluants avec la préférence $Ble <_{PA} *$

téressent l'utilisateur. Supposant maintenant que l'utilisateur souhaite se focaliser sur un type de culture bien précis, la culture $RGA\text{-}TBmoins$. Il souhaite spécifier des préférences sur les valeurs spécialisées de cette culture en formulant des préférences entre les différents ITK de la culture $RGA\text{-}TBmoins$. Grâce à la méthode $HSky$, ce type de requête associée à des préférences spécialisées ne nécessite pas le recalcul de tous les ITK appartenant au skyline.

Dans la table 6.6, nous décrivons le résultat correspondant à la requête suivante :

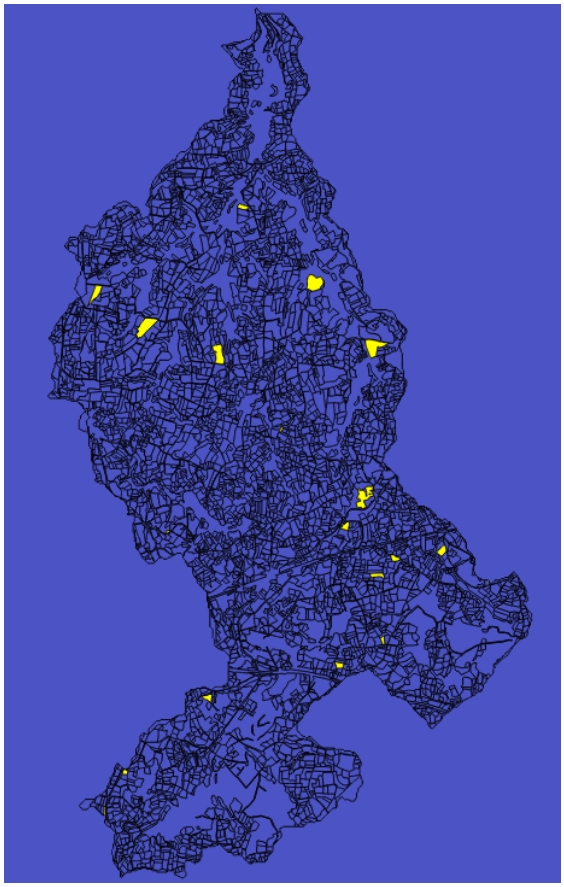


FIGURE 6.1 – Les parcelles les moins polluées avec la préférence $Prairie - p <_{PA} *$

ID ITK	Type ITK	Culture	DNRU (Kg/Ha/Année)	Durée (j)
9	blé-p-cipan-moutarde	blé	0,02	300
205	cipan-moutarde	CIPAN	0,09	188
287	maïs-e-FV[3]	maïs	3,24	180
566	pdtFB[4]	PDT	0,54	121
678	prairie-p	prairie-p	0,46	165
1671	RGA-TBmoins[5]	RGA-TBmoins	15,18	150
2446	RGA-TBmoins-aut[6]	RGA-TBmoins	80,4	30
1407	RGA-LPaut[7]	prairie-t	100,1	30

TABLE 6.4 – Les ITK les plus polluants avec la préférence $Ble <_{PA} *$

*Pour les itinéraires techniques de Blé et de type RGA-TBmoins-aut*1-2007*, quels sont ceux qui polluent le plus ?*

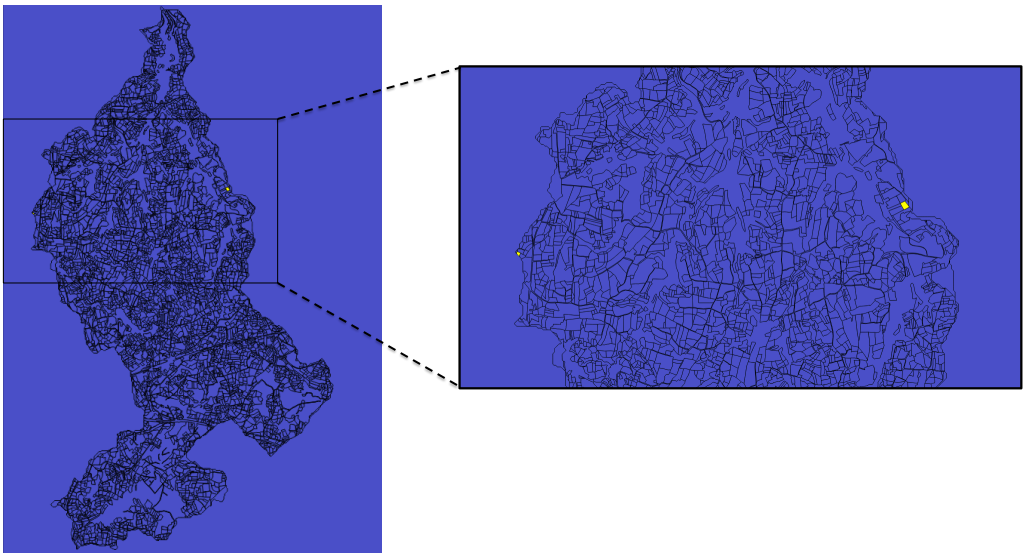


FIGURE 6.2 – Les parcelles les plus polluées avec la préférence $Prairie - p <_{PA} *$

ID ITK	Type ITK	Culture	DNRU (Kg/Ha/Année)	Durée (j)
9	blé-p-cipan-moutarde	blé	0,02	300
1671	RGA-TBmoins	RGA-TBmoins	15,18	150
2446	RGA-TBmoins-aut	RGA-TBmoins	80,4	30
1407	RGA-LPaut	prairie-t	100,1	30

TABLE 6.5 – Les ITK les plus polluants avec la préférence $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$

La préférence exprimée sur la dimension *Pratique Agricole* correspond à la préférence suivante : $Ble <_{PA} RGA - TBmoins <_{PA} *$, $RGA - TBmoins - aut1 - 2007 <_{PA} RGA - TBmoins$. Cette préférence est une spécialisation de la préférence $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$ (cf. table 6.6). Pour calculer les ITK associés à la préférence $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$, $RGA\text{-}TBmoins\text{-}aut1\text{-}2007 <_{PA} RGA - TBmoins$, nous devons tout d'abord recalculer la requête relative à la préférence $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$ avec l'algorithme $HSky$ afin de construire la structure de spécialisation / généralisation associée à cette préférence. Le coût en terme de temps de pré-traitement de cette structure est égal à 35 *sec.* Une fois la structure construite, l'utilisateur a la possibilité d'affiner ou de généraliser ses préférences, et peut donc extraire les ITK associés à la spécialisation $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$, $RGA\text{-}TBmoins\text{-}aut1\text{-}2007 <_{PA} RGA - TBmoins$. Pour cette préférence, l'algorithme $HSky$ a retourné 7 ITK (cf. table 6.3). L'ITK 1671 est disqualifié de l'ensemble skyline car il est dominé par l'ITK 2446. Le coût en terme de temps de réponse à cette requête est de l'ordre de millièmes de seconde. Cela souligne l'efficacité de notre méthode $HSky$ dans le cadre

de l'application *N-Catch*. La structure de *HSky* implique un coût en terme de temps de pré-traitement. Pour que l'utilisation de cette structure soit bénéficiaire, il faut un certain nombre d'interactions utilisateur.

ID ITK	Type ITK	Culture	DNRU (Kg/Ha/Année)	Durée (j)
9	blé-p-cipan-moutarde	Blé	0,02	300
205	cipan-moutarde	CIPAN	0,09	188
287	maïs-e-FV	maïs	3,24	180
566	pdtFB	PDT	0,54	121
678	prairie-p	Prairie-p	0,46	165
2446	RGA-TBmoins-aut	RGA-TBmoins	80,4	30
1407	RGA-LPaut	Prairie-t	100,1	30

TABLE 6.6 – Les ITK les plus polluants avec la préférence $Ble <_{PA} RGA\text{-}TBmoins <_{PA} *$, $RGA\text{-}TBmoins\text{-}aut\text{-}2007 <_{PA} RGA - TBmoins$

6.3 Conclusion

Dans ce chapitre, nous avons appliqué les algorithmes EC^2Sky et $HSky$ sur les résultats de simulations stockés dans l'entrepôt de données *N-Catch* décrit dans le chapitre 3. Nous avons montré comment exploiter les préférences des utilisateurs afin de détecter les données susceptibles de les intéresser, et de trouver les solutions compromis associées à leurs attentes. Un ensemble de questions thématiques, auxquelles les requêtes skyline associées à des préférences dynamiques ou hiérarchiques couplées avec l'entrepôt de données *N-Catch* peuvent apporter des réponses, a été constitué. Les expérimentations réalisées sur une partie de ces questions soulignent l'efficacité des algorithmes EC^2Sky et $HSky$, et montrent que les requêtes skyline permettent de pallier le problème de la rigidité des requêtes $OLAP$ et contribuent à l'amélioration du pouvoir de discrimination de ces dernières.

Les utilisateurs ont aussi formulé des questions sollicitant des données supplémentaires au niveau de l'entrepôt de données *N-Catch*. En particulier, il s'agit d'enrichir la dimension spatiale en y ajoutant des niveaux hiérarchiques (ex. sous-bassin, exploitation, ...etc) afin d'augmenter les possibilités de navigation et d'agrégation spatiales. Ces questions constituent de bonnes perspectives d'enrichissement de l'application *N-Catch* pour une meilleure exploitation des algorithmes développés. Dans le chapitre suivant, nous établissons un bilan des contributions apportées et nous traçons différentes perspectives de recherche qui pourraient être menées ultérieurement, tant du point de vue théorique, que du cas traité.

Conclusion

Dans cette thèse, nous nous sommes intéressés à élaborer des méthodes d'analyse de résultats de simulation qui replacent l'utilisateur au coeur du processus décisionnel, et qui permettent d'analyser et d'interpréter de gros volumes de données de manière efficace.

La démarche proposée consiste à utiliser des méthodes d'analyse multidimensionnelle interactive. Tout d'abord, nous avons proposé une méthode d'archivage des résultats de simulation dans une base de données décisionnelle (i.e. entrepôt de données), adaptée au caractère spatio-temporel des données de simulation produites. Ensuite, nous avons suggéré d'analyser ces données de simulations avec des méthodes d'analyse en ligne (OLAP) afin de fournir aux acteurs des informations stratégiques pour améliorer le processus d'aide à la prise de décision.

Enfin, nous avons proposé deux méthodes d'extraction de skyline dans le contexte des entrepôts de données afin de permettre aux utilisateurs de formuler de nouvelles questions en combinant des critères environnementaux contradictoires, et de trouver les solutions compromis associées à leurs attentes, puis d'exploiter les préférences des utilisateurs pour détecter et faire ressortir les données susceptibles de les intéresser. La première méthode EC^2Sky, permet un calcul incrémental et efficace des skyline en présence de préférences utilisateurs dynamiques, et ce malgré de gros volumes de données. La deuxième méthode $HSky$, étend la recherche des points skyline aux dimensions hiérarchiques. Elle permet aux utilisateurs de naviguer le long des axes des dimensions hiérarchiques (i.e. spécialisation / généralisation) tout en assurant un calcul en ligne des points skyline correspondants.

Ces contributions ont été motivées et expérimentées par l'application de gestion des pratiques agricoles pour l'amélioration de la qualité des eaux des bassins versants agricoles, et nous avons proposé un couplage entre le modèle d'entrepôt de données agro-hydrologiques construit et les méthodes d'extraction de skyline proposées.

Bilan des contributions

Une première contribution concerne la proposition d'une méthodologie de construction d'un modèle d'entrepôt de données agro-hydrologique $N\text{-}Catch$, dont les principales étapes représentent : le pré-traitement et la transformation des données issues d'un modèle agro-hydrologique, la modélisation multidimensionnelle et hiérarchique des pratiques agricoles, et l'analyse des résultats de simulations en combinant la modélisa-

tion spatio-temporelle des données, l'entreposage des données et l'analyse en ligne. Ces contributions méthodologiques peuvent être appliquées à une variété de problématiques agro-environnementales. Plus généralement, cette approche peut être utilisée pour analyser l'impact des pratiques agricoles sur la qualité des eaux, ou d'autres impacts environnementaux, et pour faciliter le processus d'aide à la décision (i.e. rendre les données facilement accessibles par les décideurs).

Une deuxième contribution concerne la proposition d'une nouvelle méthode efficace de calcul incrémental de skyline en présence de dimensions associées à des préférences dynamiques : EC^2Sky. Ainsi, sur certaines dimensions l'ordre peut changer dynamiquement. Nous avons étudié les préférences sur des valeurs de dimensions qui peuvent être exprimées par un ordre total ou partiel, avec une attention particulière portée aux points compromis qui sont essentiels pour une bonne prise de décision. Notre approche est basée sur une matérialisation des préférences de premier ordre, qui permet de répondre de manière efficace aux requêtes de type skyline en présence de préférences utilisateurs malgré de gros volumes de données. Les expérimentations réalisées dans cette étude soulignent la pertinence de la solution proposée EC^2Sky en la comparant aux références du domaine.

Une troisième contribution concerne la proposition d'une nouvelle méthode $HSky$ qui étend la recherche des points skyline aux dimensions hiérarchiques. La définition de hiérarchies sur les dimensions permet à l'utilisateur d'affiner ou de généraliser ses préférences lors du calcul des skyline. Nous avons mis en évidence un ensemble de propriétés permettant la formalisation des relations hiérarchiques entre les préférences associées aux différentes dimensions. Ces propriétés permettent notamment aux utilisateurs de naviguer le long des relations hiérarchiques entre les préférences tout en assurant un calcul en ligne des points skyline correspondants. En particulier, la propriété de monotonie hiérarchique a contribué considérablement à l'analyse et au calcul des points skyline en présence de préférences associées à des dimensions hiérarchiques. En effet, cette propriété indique que tout point skyline associé à une préférence donnée reste skyline lorsque nous considérons une généralisation de cette préférence. Cette propriété nous a permis de caractériser les points skyline qui restent skyline et ceux qui ne le sont plus, lors de la spécialisation (drill-down) ou de la généralisation (roll-up) des préférences associées aux dimensions hiérarchiques. Ceci nous a permis (i) de réduire le nombre de tests de dominance pour calculer efficacement les skyline lors de la navigation sur les dimensions hiérarchiques, et (ii) de proposer la structure $HSky$ qui rend possible la navigation le long des relations hiérarchiques entre les préférences. Les expérimentations réalisées dans cette étude soulignent l'efficacité des performances de $HSky$ dans un contexte d'analyse en ligne.

Afin de démontrer l'utilité de des méthodes EC^2Sky et $HSky$, nous avons appliqué ces algorithmes sur les données de simulations issues du modèle TNT, dans le cadre de l'analyse multidimensionnelle des données stockées dans l'entrepôt de données $N\text{-}Catch$. Nous avons montré comment exploiter les préférences des utilisateurs afin d'extraire les données susceptibles de les intéresser, et de trouver les solutions compromis associées à leurs attentes. Un ensemble de questions thématiques a été constitué. Nous avons montré comment associer une requête skyline avec des préférences dynamiques ou hié-

rarchiques à chacune de ces questions et présenté les résultats de ces requêtes sur l'entrepôt *N-Catch*. Les expérimentations réalisées sur ces questions soulignent l'efficacité des performances des algorithmes EC^2Sky et $HSky$ dans le contexte de l'application traitée.

Perspectives

Les travaux de recherche menés dans le cadre de cette thèse ouvrent plusieurs pistes de recherche prometteuses.

Enrichissement de l'entrepôt de données. Un des premiers intérêts de l'application *N-Catch* est de permettre une analyse spatio-temporelle des résultats de simulation issus du modèle TNT. Afin de parvenir à quantifier l'impact des pratiques agricoles sur la pollution nitrique à différentes échelles spatio-temporelles. Cependant, comme les données spatiales disponibles au moment de la construction de *N-Catch* se limitait à l'échelle de la parcelle et du bassin versant, il est alors difficile pour l'utilisateur d'exploiter cette dimension dont la hiérarchie est très restreinte. Pourtant, le modèle TNT considère plusieurs granularités spatiales (ex. sous-bassin, exploitation, type de sols,...). Un travail intéressant serait d'enrichir la hiérarchie de la dimension spatiale avec les nouvelles données disponibles, et d'offrir à l'utilisateur de nouvelles possibilités de navigation spatiales.

Il serait aussi intéressant d'envisager plusieurs simulations issues du modèle TNT, en ajoutant une dimension *Simulation* dans *N-Catch*, afin de permettre l'archivage de plusieurs simulations. Ceci revient à ajouter un nouvel axe d'analyse qui permet de comparer les différents scénarios simulés par TNT, mais aussi d'analyser l'évolution des sorties du modèle lorsque les entrées varient d'une certaine façon.

Implication de l'utilisateur dans le processus décisionnel. L'un des principaux objectifs de cette thèse est de replacer l'utilisateur au coeur du processus décisionnel. Nous avons proposé des méthodes d'analyse qui prennent en compte les préférences des utilisateurs. Nous pensons qu'il serait intéressant de développer une interface utilisateur simple et intuitive qui centralise et facilite l'accès à toutes les méthodes développées au cours de cette thèse. Nous avons proposé dans le chapitre 3 de coupler l'application *N-Catch* avec le système d'information géographique *QGis*, et nous avons montré l'intérêt d'une visualisation cartographique des différentes données stockées dans *N-Catch*. L'utilisation de la carte comme outil d'exploration et de visualisation de données permet d'avoir un modèle décisionnel se rapprochant davantage de la réalité de l'utilisateur et lui demandant un moins grand effort d'abstraction, ce qui accroît son efficacité. Il serait pertinent de proposer un couplage des deux algorithmes d'extraction de skyline (EC^2Sky et $HSky$) avec *QGis* afin de permettre une visualisation cartographique des skyline retournés.

À ce jour l'exploitation des résultats issus du modèle TNT est restreinte à ses concepteurs et à quelques chercheurs agronomes. Or l'idée est de rendre l'exploitation

de ces données accessible par d'autres utilisateurs, par exemple à des gestionnaires de bassins versants,etc. De plus, le temps imparti n'a pas permis une interaction approfondie avec les chercheurs agronomes sur la pertinence des résultats obtenus. Pour cela, une nouvelle approche, basée sur le dialogue avec les utilisateurs potentiels, est nécessaire afin d'identifier leurs besoins et leurs attentes.

Toujours dans l'optique d'améliorer l'interaction de l'utilisateur avec les méthodes d'analyse développées dans cette thèse, nous pensons qu'il serait intéressant de gérer la partie interactive du processus d'analyse au fur et à mesure de l'acquisition des connaissances. Par exemple, un utilisateur peut poser une requête $OLAP$ sur les données stockées dans $N\text{-}Catch$. Une fois les résultats de cette requête analysés, il serait pertinent de donner à l'utilisateur la possibilité de raffiner ces résultats en y appliquant une nouvelle requête ($OLAP$ ou skyline) et ce de manière interactive et efficace (i.e. calcul en ligne).

Extraction de skyline en présence de dimensions hiérarchiques associées à des préférences dynamiques Dans la méthode $HSky$ détaillée au chapitre 5, nous proposons à l'utilisateur de naviguer le long des spécialisations / généralisations d'une préférence de base définie au préalable. Cependant, cette méthode restreint l'utilisateur à ne matérialiser que les préférences hiérarchiques relatives à la préférence de base. Il serait trop coûteux, en terme de temps de pré-calcul et de stockage mémoire, de stocker toutes les spécialisations/ généralisations de toutes les préférences possibles avec la structure $HSky$. Nous pensons que le couplage des deux méthodes EC^2Sky et $HSky$ peut répondre à cette problématique. L'idée est de ne matérialiser que les préférences hiérarchiques de premier ordre comme proposé dans la méthode EC^2Sky chapitre 4, tout en maintenant les liens de spécialisation / généralisations entre les préférences. Ce couplage combine les avantages des deux méthodes EC^2Sky et $HSky$. En effet, il permet un calcul incrémental des skyline associés à un ensemble de préférences, et facilite la modification interactive de ces dernières. Il fournit aussi à l'utilisateur la possibilité d'affiner ou de généraliser ses préférences lors du calcul des skyline.

Optimisation de la navigation au sein de la structure des préférences hiérarchiques La structure de spécialisations / généralisations des préférences hiérarchiques construite par la méthode $HSky$ peut être considérée comme un cube de préférences en analogie au cube de données. Chaque noeud de la structure $HSky$ représente un cuboïde (i.e. cellule du cube) décrit par ses préférences. Nous pensons alors qu'il serait intéressant de s'inspirer des différents travaux portant sur la matérialisation partielle des cubes de données [HMT09] afin d'optimiser l'évaluation des requêtes skyline sans dépasser une limite d'espace mémoire. Cela revient à caractériser les cellules du cube de préférences à matérialiser en se basant sur les propriétés relatives aux requêtes skyline. Les motivations de ce travail consistent à faciliter et à optimiser la navigation au sein de la structure $HSky$, et à réduire l'espace mémoire requis par le stockage de cette structure.

Augmenter l'expressivité des requêtes skyline Cette voie de recherche vise à enrichir et à étendre les types de préférences définis dans EC^2Sky et $HSky$ par l'introduction de la conditionnalité. En pratique, il est courant que l'expression des préférences dépende du contexte et du profil de l'utilisateur et fasse intervenir des priorités. Cependant, les cadres proposés jusqu'à présent dans le contexte des requêtes skyline ne prennent pas en considération la notion de préférences conditionnelles : si la condition C est vraie alors préférer A à B. En revanche, ce type de préférences a été largement étudié dans la communauté $Intelligence\ Artificielle$, pour exemple, les réseaux de préférences $CP\text{-}Nets$ ($Conditional\ Preferences\ Networks$ [BBD$^+$04]). Dans le cadre de l'application $N\text{-}Catch$, nous pensons pertinent de pouvoir adapter ses préférences selon le profil utilisateur ou selon le scénario simulé. Par exemple, pour un scénario simulant des systèmes herbagers, il est plus pertinent de préférer des parcelles de type $Prairies\ Permanentes$ au reste des parcelles. De manière générale, un travail intéressant serait d'étudier l'évolution des skyline en présence de telles préférences.

Bibliographie

[Abd09] A. Abdullah. Analysis of mealybug incidence on the cotton crop using ADSS-OLAP (Online Analytical Processing) tool. *Computers and Electronics in Agriculture*, 69(1) :59–72, 2009.

[AWAA09] S. Antony, P. Wu, D. Agrawal, and A. E. Abbadi. Aggregate skyline : Analysis for online users. In *Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet*, pages 50–56. IEEE Computer Society, 2009.

[AWAEA08] S. Antony, P. Wu, D. Agrawal, and A. El Abbadi. Moolap : Towards multi-objective olap. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 1394–1396. IEEE Computer Society, 2008.

[BBCX+10] Ding B., Zhao B., Lin C. X., Han J., and Zhai C. Topcells : Keywordbased search of top-k aggregated documents in text cube. In *In International Conference on Data Engineering (ICDE*, 2010.

[BBD+04] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets : a tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Int. Res.*, 21(1) :135–191, 2004.

[BCQ12] T. Bouadi, M. Cordier, and R. Quiniou. Incremental computation of skyline queries with dynamic preferences. In *DEXA (1)*, pages 219–233, 2012.

[BCQ13] T. Bouadi, M-.O. Cordier, and R. Quiniou. Computing skyline incrementally in response to online preference modification. *T. Large-Scale Data- and Knowledge-Centered Systems*, 10 :34–59, 2013.

[BCQG11] T. Bouadi, M.-O. Cordier, R. Quiniou, and C. Gascuel. Analyse multidimensionnelle interactive de résultats de simulation d'un agro-système. In *INFORSID*, 2011.

[BDR01] V. Beaujouan, P. Durand, and L. Ruiz. Modelling the effect of the spatial distribution of agricultural practices on nitrogen fluxes in rural catchments. *Ecological modelling*, 137(1) :93–105, 2001.

[BDR+02] V. Beaujouan, P. Durand, L. Ruiz, P. Aurousseau, and G. Cotteret. A hydrological model dedicated to topography-based simulation of nitrogen transfer and transformation : rationale and application to the

geomorphology ? denitrification relationship. *Hydrological Processes*, 16(2) :493–507, 2002.

[BF09a] F. Bentayeb and C. Favre. Rok : Roll-Up with the K-Means clustering method for recommending OLAP queries. In *Proceedings of the 20th International Conference on Database and Expert Systems Applications*, pages 501–515. Springer-Verlag, 2009.

[BF09b] F. Bentayeb and C. Favre. Rok : Roll-up with the k-means clustering method for recommending olap queries. In *Proceedings of the 20th International Conference on Database and Expert Systems Applications*, pages 501–515. Springer-Verlag, 2009.

[BGG07] C. Brando, M. Goncalves, and V. González. Evaluating top-k skyline queries over relational databases. In *Proceedings of DEXA'07*, pages 254–263, 2007.

[BGM+05] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for olap queries. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 9–18. ACM, 2005.

[BGOC+] T. Bouadi, C. Gascuel-Odoux, M.-O. Cordier, R. Quiniou, P. Moreau, and J. Salmon.-Monviola. N-catch : A data warehouse for multilevel analysis of simulated nitrogen data from an agro-hydrological model. *Computers and Electronics in Agriculture*, xx(xx) :xx, *en cours de soumission.*

[BGOC+13] T. Bouadi, C. Gascuel-Odoux, M.-O. Cordier, R. Quiniou, and P. Moreau. Using data warehouses to extract knowledge from agro-hydrological simulations. In *EGU General Assembly Conference Abstracts*, volume 15, page 9322, 2013.

[BGS06] W.T. Balke, U. Guntzer, and W. Siberski. Exploiting indifference for customization of partial order skylines. In *Proceedings of the 10th International Database Engineering and Applications Symposium*, pages 80–88. IEEE Computer Society, 2006.

[Bim07] S. Bimonte. *Intégration de l'information géographique dans les entrepôts de données et l'analyse en ligne : de la modélisation à la visualisation.* Thèse de doctorat en informatique, INSA de Lyon, 2007.

[BKS01] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc of the 17th International Conference on Data Engineering*, pages 421–430. IEEE Computer Society, 2001.

[BKST78] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM*, pages 536–543, 1978.

[BM79] G. R. Bitran and T. L. Magnanti. The structure of admissible points with respect to cone dominance. *Optimization Theory and Applications*, pages 573–614, 1979.

[BMBLR07] R. Ben Messaoud, O. Boussaid, and S. Loudcher Rabaséda. A multiple correspondence analysis to organize data cubes. In *Proceedings of the 2007 conference on Databases and Information Systems IV : Selected Papers from the Seventh International Baltic Conference DB&IS'2006*, pages 133–146, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[BMH01] Y. Bédard, T. Merret, and J. Han. Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery. *Data Mining and Knowledge Discovery*, 2001.

[BMR+98] N. Brisson, B. Mary, D. Ripoche, M.H. Jeuffroy, F. Ruget, B. Nicoullaud, P Gate, F. Devienne-Barret, R. Antonioletti, C. Durr, G. Richard, N. Beaudoin, S. Recous, X. Tayot, D. Plenet, P. Cellier, J.M. Machet, J.M. Meynard, and R. Delécolle. STICS : a generic model for the simulation of crops and their water and nitrogen balances. i. theory and parameterization applied to wheat and corn. *Agronomie*, 18(1) :311–346, 1998.

[BNBMLRB08] A. Bodin-Niemczuk, R. Ben Messaoud, S. Loudcher Rabaséda, and O. Boussaid. Vers l'intégration de la prédiction dans les cubes OLAP. In *Proceedings of EGC'08*, pages 203–204, 2008.

[BRP06] Y. Bédard, S. Rivest, and M. Proulx. Spatial On-Line Analytical Processing (SOLAP) : Concepts, Architectures, and Solutions from a Geomatics Engineering Perspective. In *Data Warehouses and OLAP : Concepts, Architecture, and*, 2006.

[CCS93] E. F. Codd, S. B. Codd, and C. T. Salley. Providing olap (on-line analytical processing) to user-analysts : An it mandate. 1993.

[CDH00] Q. Chen, U. Dayal, and M. Hsu. An olap-based scalable web access analysis engine. In *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery*, pages 210–223. Springer-Verlag, 2000.

[CGG+05] M.-O. Cordier, F. Garcia, C. Gascuel, V. Masson, J. Salmon-Monviola, F. Tortrat, and R. Trépos. A machine learning approach for evaluating the impact of land use and management practices on streamwater pollution by pesticides. In Modelling, Simulation Society of Australia, and New Zealand, editors, *MODSIM'05 (International Congress on Modelling and Simulation)*, 2005.

[CGGL03] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. *IEEE Trans. on Knowl. and Data Eng.*, pages 717– 719, 2003.

[CGGL05] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting : Theory and optimizations. In *In Proc of Intelligent Information Systems*, pages 595–604. Springer Berlin/Heidelberg, 2005.

[Cha98] Surajit Chaudhuri. Data mining and database systems : Where is the intersection ? *IEEE Data Eng. Bull.*, 21(1) :4–8, 1998.

[CRST06] B.-C. Chen, R. Ramakrishnan, J. W. Shavlik, and P. Tamma. Bellwether analysis : predicting global aggregates from local regions. In *Proceedings of the 32nd international conference on Very large data bases*, pages 655–666. VLDB Endowment, 2006.

[CZC01] M. Chen, Q. Zhu, and Z. Chen. An integrated interactive environment for knowledge discovery from heterogeneous data resources. *Information & Software Technology*, 43(8) :487–496, 2001.

[Deb08] K. Deb. A robust evolutionary framework for multi-objective optimization. In *Proc of the 10th annual conference on Genetic and evolutionary computation*, pages 633–640. ACM, 2008.

[DFS02] Kossmann D., Ramsak F., and Rost S. Shooting stars in the sky : An online algorithm for skyline queries. In *Very Large Data Bases*, pages 275–286, 2002.

[DGOC02] P. Durand, C. Gascuel-Odoux, and M.-O. Cordier. Parameterisation of hydrological models : a review and lessons learned from studies of an agricultural catchment (Naizin, France). *Agronomie*, 22(2) :217–228, 2002.

[Gar97] Kenn Gardels. Open gis and on-line environmental libraries. *SIGMOD Rec.*, 26(1) :32–38, 1997.

[GBLP96] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube : A relational aggregation operator generalizing Group-By, cross-tab, and Sub-Total. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 152–159. IEEE Computer Society, 1996.

[GC98] S. Goil and A. Choudhary. High performance multidimensional analysis and data mining. In *Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–2. IEEE Computer Society, 1998.

[GC01] S. Goil and A. Choudhary. PARSIMONY : An infrastructure for parallel multidimensional analysis and data mining. *J. Parallel Distrib. Comput.*, 61(3) :285–321, 2001.

[GMN08] A. Giacometti, P. Marcel, and E. Negre. A framework for recommending OLAP queries. In *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP*, pages 73–80. ACM, 2008.

[GOAC+09a] C. Gascuel-Odoux, P. Aurousseau, M-.O. Cordier, P. Durand, F. Garcia, V. Masson, J. Salmon-Monviola, F. Tortrat, and R. Trepos. A decision-oriented model to evaluate the effect of land use and agricultural management on herbicide contamination in stream water. *Environmental Modelling and Software*, 24(12) :1433–1446, 2009.

[GOAC+09b] C. Gascuel Odoux, Pierre Aurousseau, Marie-Odile Cordier, Patrick Durand, Frederick Garcia, Véronique Masson, Jordy Salmon-Monviola, Florent Tortrat, and Ronan Trépos. A decision-oriented model to evaluate the effect of land use and agricultural management on herbicide

contamination in stream water. *Environmental modelling & software*, pages 1433–1446, 2009.

[GR09] M. Golfarelli and S. Rizzi. Expressing olap preferences. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, pages 83–91. Springer-Verlag, 2009.

[GS06] A.K. Gupta and A. I. Sivakumar. Pareto control in multi-objective dynamic scheduling of a stepper machine in semiconductor wafer fabrication. In *Proc of the 38th conference on Winter simulation*, pages 1749–1756. Winter Simulation Conference, 2006.

[GSG05] P. Godfrey, R. Shipley, and J. Gryz. Maximal vector computation in large data sets. In *Proceedings of the 31st international conference on Very large data bases*, pages 229–240. VLDB Endowment, 2005.

[HAC+99] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive data analysis : The control project. *Computer*, 32(8) :51–59, 1999.

[Han97] J. Han. Olap mining : An integration of OLAP with data mining. In *In Proceedings of the 7th IFIP 2.6 Working Conference on Database Semantics*, pages 1–9, 1997.

[Han98] J. Han. Towards on-line analytical mining in large databases. *SIGMOD Rec.*, 27(1) :97–107, 1998.

[HFZT08] S. Hackel, M. Fischer, D. Zechel, and T. Teich. A multi-objective ant colony approach for pareto-optimization using dynamic programming. In *Proc of the 10th annual conference on Genetic and evolutionary computation*, pages 33–40. ACM, 2008.

[HG00] C. Hénault and J. C. Germon. NEMIS, a predictive model of denitrification on the field scale. *European Journal of Soil Science*, 51(2) :257–270, 2000.

[HGSW08] Z. Huang, J. Guo, S.L. Sun, and W. Wang. Efficient optimization of multiple subspace skyline queries. *J. Comput. Sci. Technol.*, pages 103–111, 2008.

[HMT09] N. Hanusse, S. Maabout, and R. Tofan. Matérialisation Partielle des Cubes de Données. In *Actes des 25èmes journées "Bases de Données Avancées" BDA'09*, pages 1–20, 2009.

[IBM98] IBM. *Data modeling techniques for data warehousing.* IBM Corp., Riverton, NJ, USA, 1998.

[IKA02] T. Imieliński, L. Khachiyan, and A. Abdulghani. Cubegrades : Generalizing association rules. *Data Min. Knowl. Discov.*, 6(3) :219–257, 2002.

[Inm92] W.H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, Inc., 1992.

[JAWCXH07] Pei J., Fu A. W.-C., Lin X., and Wang H. Computing compressed multidimensional skyline cubes efficiently. In *ICDE'07*, pages 96–105, 2007.

[JGO02] L. Jinho, D. Grossman, and R. Orlandic. Mire : a multidimensional information retrieval engine for structured data and text. In *Information Technology : Coding and Computing, 2002. Proceedings. International Conference on*, pages 224–229, 2002.

[JRTZ08] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. Management of context-aware preferences in multidimensional databases. In *Digital Information Management, 2008. ICDIM 2008. Third International Conference on*, pages 669–675, 2008.

[JTEH07] W. Jin, A. K. H. Tung, M. Ester, and J. Han. On efficient processing of subspace skyline queries on high dimensional data. In *Proc of the 19th International Conference on Scientific and Statistical Database Management*, page 12. IEEE Computer Society, 2007.

[LB12] S. Loudcher and O. Boussaid. Olap on complex data : Visualization operator based on correspondence analysis. In *IS Olympics : Information Systems in a Diverse World*, volume 107, pages 172–185. Springer Berlin Heidelberg, 2012.

[LCB+11] C. Largouët, M.-O. Cordier, Y.-M. Bozec, Y. Zhao, and G. Fontenelle. Use Of Timed Automata And Model-Checking To Explore Scenarios On Ecosystem Models. *Environmental Modelling and Software*, (30) :123–138, 2011.

[LGMR05] P.A. Longley, M.F. Goodchild, D.J. Maguire, and D.W. Rhind. *Geographic Information Systems and Science*. John Wiley & Sons, 2005.

[LOTW06] C. Li, B. C. Ooi, A. K. H. Tung, and S. Wang. Dada : a data cube for dominant relationship analysis. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 659–670. ACM, 2006.

[Lou11] S. Loudcher. *Vers l'OLAP sémantique pour l'analyse en ligne des données complexes*. PhD thesis, 2011.

[MBR06] R.B. Messaoud, O. Boussaid, and S.L. Rabaséda. Using a factorial approach for efficient representation of relevant olap facts. In *Databases and Information Systems, 2006 7th International Baltic Conference on*, pages 98–105, 2006.

[MC11] D. Mindolin and J. Chomicki. Preference elicitation in prioritized skyline queries. *The VLDB Journal*, 20(2) :157–182, 2011.

[MCDA03] J. Mothe, C. Chrisment, B. Dousset, and J. Alaux. Doccube : multi-dimensional visualisation and exploration of large document sets. *J. Am. Soc. Inf. Sci. Technol.*, 54(7) :650–659, 2003.

[MHC97] Kamber M., J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *In Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97*, pages 207–210, 1997.

[MI13] Magnani M. and Assent I. From stars to galaxies : skyline queries on aggregate data. In *EDBT*, pages 477–488, 2013.

[Mie05] I. Mierswa. Incorporating fuzzy knowledge into fitness : multiobjective evolutionary 3d design of process plants. In *GECCO*, pages 1985–1992. ACM, 2005.

[MLC+00] M. C. McCabe, J. Lee, A. Chowdhury, D. Grossman, and O. Frieder. On the design and evaluation of a multi-dimensional approach to information retrieval (poster session). In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 363–365. ACM, 2000.

[MPJ07] M. Morse, J. M. Patel, and H. V. Jagadish. Efficient skyline computation over low-cardinality domains. In *Proceedings of the 33rd international conference on Very large data bases*, pages 267–278. VLDB Endowment, 2007.

[MRBB04] Ri. B. Messaoud, S. Rabaséda, O. Boussaid, and F. Bentayeb. Opac : opérateur d'analyse en ligne basé sur une technique de fouille de données. In Georges Hébrail, Ludovic Lebart, and Jean-Marc Petit, editors, *EGC*, volume RNTI-E-2 of *Revue des Nouvelles Technologies de l'Information*, pages 35–46. Cépaduès-Éditions, 2004.

[MRM+12] P. Moreau, L. Ruiz, F. Mabon, T. Raimbault, P. Durand, L. Delaby, S. Devienne, and F. Vertès. Reconciling technical, economic and environmental efficiency of farming systems in vulnerable areas. *Agriculture, Ecosystems & Environment*, 147(0) :89 – 99, 2012.

[MRR+12] P. Moreau, L. Ruiz, T. Raimbault, F. Vertès, M.O. Cordier, C. Gascuel-Odoux, V. Masson, J. Salmon-Monviola, and P. Durand. Modeling the potential benefits of catch-crop introduction in fodder crop rotations in a western europe landscape. *Science of The Total Environment*, 437(0) :276–284, 2012.

[NQN04] S. Naouali, M. Quafafou, and G. Nachouki. Mining olap cubes : semantic links based on frequent itemsets. In *Information and Communication Technologies : From Theory to Applications, 2004. Proceedings. 2004 International Conference on*, pages 447–449, 2004.

[NSR08] S. Nilakanta, K. Scheibe, and A. Rai. Dimensional issues in agricultural data warehouse designs. *Computers and Electronics in Agriculture*, 60(2) :263–278, 2008.

[PHP+01] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth*

international conference on Information and knowledge management, pages 81–88. ACM, 2001.

[PJET05] J. Pei, W. Jin, M. Ester, and Y. Tao. Catching the best views of skyline : a semantic approach based on decisive subspaces. In *Proc of the 31st international conference on Very large data bases*, pages 253–264. VLDB Endowment, 2005.

[PLL+10] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, and Y.W. Choong. Mining multidimensional and multilevel sequential patterns. *ACM Trans. Knowl. Discov. Data*, pages 4 :1–4 :37, 2010.

[PMB+10] F. Pinet, A. Miralles, S. Bimonte, F. Vernier, N. Carluer, V. Gouy, and S. Bernard. The use of UML to design agricultural data warehouses. In *AgEng 2010, International Conference on Agricultural Engineering*, 2010.

[PP03] T. Priebe and G. Pernul. Towards integrative enterprise knowledge portals. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 216–223. ACM, 2003.

[PTFS03] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *Proc of the 2003 ACM SIGMOD international conference on Management of data*, pages 467–478. ACM, 2003.

[PTFS05] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, pages 41–82, 2005.

[RPK10] C. Raïssi, J. Pei, and T. Kister. Computing closed skycubes. *Proc. VLDB Endow.*, pages 838–847, 2010.

[RT09] F. Ravat and O. Teste. Personalization and olap databases. In Stanislaw Kozielski and Robert Wrembel, editors, *New Trends in Data Warehousing and Data Analysis*, volume 3, pages 1–22. Springer US, 2009.

[SAM98] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proceedings of the 6th International Conference on Extending Database Technology : Advances in Database Technology*, pages 168–182. Springer-Verlag, 1998.

[Sar99] S. Sarawagi. Explaining differences in multidimensional aggregates. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 42–53. Morgan Kaufmann Publishers Inc., 1999.

[Sar01] S. Sarawagi. idiff : Informative summarization of differences in multidimensional aggregates. *Data Min. Knowl. Discov.*, 5(4) :255–276, 2001.

[SMDF+12] J. Salmon-Monviola, P. Durand, F. Ferchaud, F. Oehler, and L. Sorel. Modelling spatial dynamics of cropping systems to assess agricultural

practices at the catchment scale. *Computers and Electronics in Agriculture*, 81(0) :1 – 13, 2012.

[SNT85] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization*. Academic Press, Orlando, 1985.

[SS01] G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional OLAP data. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 531–540. Morgan Kaufmann Publishers Inc., 2001.

[Ste86] R.E. Steuer. *Multiple criteria optimization : Theory computation and applications*. New York : John Wiley & Sons, Inc., 1986.

[SZJ07] L. Su, P. Zou, and Y. Jia. Adaptive mining the approximate skyline over data stream. In *Proc of the 7th international conference on Computational Science*, pages 742–745. Springer-Verlag, 2007.

[TD06] Xia T. and Zhang D. Refreshing the sky : the compressed skycube with efficient support for frequent updates. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 491–502. ACM, 2006.

[TEO01] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 301–310. Morgan Kaufmann Publishers Inc., 2001.

[Tre94] G. Trenkler. Univariate discrete distributions : N.l. johnson, s. kotz and a.w. kemp (1992) : 2nd edition. new york : John wiley, isbn 0-471-54897-9. *Computational Statistics & Data Analysis*, pages 240–241, 1994.

[TSAC+05] R. Trepos, A. Salleb-Aouissi, M.-O. Cordier, V. Masson, and C Gascuel-Odoux. A distance-based approach for action recommendation. In *ECML*, pages 425–436, 2005.

[TSAC+13] R. Trepos, A. Salleb-Aouissi, M.-O. Cordier, V. Masson, and C. Gascuel-Odoux. Building actions from classification rules. *Knowl. Inf. Syst.*, 34(2) :267–298, 2013.

[TXP08] Y. Tao, X. Xiao, and J. Pei. Efficient skyline and top-k retrieval in subspaces. *IEEE Trans. on Knowl. and Data Eng.*, pages 1072–1088, 2008.

[WFP+08] R. C. W. Wong, A. W. C. Fu, J. Pei, Y. S. Ho, T. Wong, and Y. Liu. Efficient skyline querying with variable user preferences on nominal attributes. *Proc. VLDB Endow.*, pages 1032–1043, 2008.

[WK06] R. Wrembel and C. Koncilia. *Data Warehouses And Olap : Concepts, Architectures And Solutions*. IRM Press, 2006.

[WMZJ07] Jin W., Ester M., Hu Z., and Han J. The multi-relational skyline operator. In *ICDE*, pages 1276–1280, 2007.

[WPFW] R. C. W. Wong, J. Pei, A. W. C. Fu, and K. Wang. An erratum on "online skyline analysis with dynamic preferences on nominal attributes". *IEEE Trans. on Knowl. and Data Eng. To be published.*

[WPFW09] R. C. W. Wong, J. Pei, A. W. C. Fu, and K. Wang. Online skyline analysis with dynamic preferences on nominal attributes. *IEEE Trans. on Knowl. and Data Eng.*, pages 35–49, 2009.

[YGJ+06] Chen Y., Dong G., Han J., Wah B. W., and Wang J. Regression cubes with lossless compression and aggregation. *IEEE Trans. Knowledge and Data Engineering*, 18 :1585–1599, 2006.

[YLL+05] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In *Proc of the 31st international conference on Very large data bases*, pages 241–252. VLDB Endowment, 2005.

[YSC+09] C. X. Yu, Y.and Lin, Y. Sun, C. Chen, J. Han, B. Liao, T. Wu, C. X. Zhai, D. Zhang, and B. Zhao. inextcube : information network-enhanced text cube. *Proc. VLDB Endow.*, 2(2) :1622–1625, 2009.

[YXJ06] Tao Y., Xiao X., and Pei J. Subsky : Efficient computation of skylines in subspaces. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 65. IEEE Computer Society, 2006.

[ZLC11] y. Zhao, C. Largouët, and M-.O. Cordier. Ecomata, un logiciel d'aide à la décision pour améliorer la gestion des écosystèmes. *Ingénierie des Systèmes d'Information*, 16(3) :85–111, 2011.

Table des figures

Liste des tableaux