République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche scientifique

Université IBN KHALDOUN - Tiaret
Faculté des Sciences et des Sciences de l'Ingénieur
Département Informatique

ÉCOLE DOCTORALE STIC

Sciences et Technologies de l'Information et de la Communication
Option : Systèmes d'Information et de Connaissance
- SIC -

Mémoire de Magister

Présenté Par :
Mohamed KENICHI

Thème

Vers une nouvelle interprétation des préférences dans les requêtes de base de données : Une approche fondée sur les CP-Nets

Jury

Président :

Youcef. DAHMANI — Maître de conférences, Université d'Ibn Khaldoun, Tiaret

Examinateurs :
Mohammed Amine. CHIKH — Maître de conférences, Université d'Abou Bekr Belkaid, Tlemcen
Abdelhafid . BESSAID — professeur, Université d'Abou Bekr Belkaid, Tlemcen

Directeur du Mémoire :

Allel HADJALI — Doctorat d'État, Maître de Conférences
IRISA/ENSSAT, Université Rennes 1

ANNEE 2010-2011

Vers une nouvelle interprétation des préférences dans les requêtes de base de données : Une approche fondée sur les CP-Nets

Résumé :

Le travail mené dans le cadre de cette mémoire repose sur une problématique centrale : L'interprétation et l'intégration des préférences d'utilisateur dans les requêtes adressées aux bases de données. En effet, La quantité d'information gérée par les systèmes de bases de données devient de plus en plus grande et il est nécessaire que les systèmes d'interrogation deviennent de plus en plus performants. Cette performance peut se mesurer en terme de temps de réponse ou en terme de qualité de l'information délivrée. Un des éléments clés de la qualité est la pertinence des réponses, en particulier, par la prise en compte des préférences des utilisateurs dans les requêtes.

La proposition dans sa globalité définit ainsi une nouvelle approche d'évaluation des requêtes à préférences basée sur les CP-Nets. L'approche proposée est fondée principalement sur trois étape : Traduire fidèlement ce que l'utilisateur désire exprimer, ces préférences et exigences ont été traduites dans un langage afin d'être ajoutées aux requêtes et exécutées. Produire un ensemble de réponses ordonnées selon le CP-Net requête déduit, Pour ce faire, nous avons étendu l'operateur ORD, Celui-ci permet d'ordonner l'ensemble des résultats d'une relation R selon la requête CP-Nets $\succ_{Q_{CP}}$. Ainsi on a proposé une nouvelle procédure adaptée pour l'évaluation de la requête CP-Nets $\succ_{Q_{CP}}$ Le résultat est une liste de tuples pertinents et ordonnés pour la requête. Enfin, La mise en œuvre de cette approche, un algorithme a été proposé pour gérer à la fois les préférences conditionnelles et le problème de la complexité du calcul des meilleures réponses sous le principe *Ceteris Paribus.*

Mot-clés : Bases de données, préférence, Représentation des préférences, complexité, Requêtes à préférences, CP-Nets, Ceteris Paribus.

Towards a novel interpretation of database preference queries: a CP-Nets based approach

Abstract:

The work conducted as part of this memory depends on a central problem: The interpretation and integration of user preferences in queries sent to databases. Indeed, the quantity of information managed by the systems of data bases becomes increasingly large and it is necessary that the systems of interrogation become increasingly powerful. This performance can be measured in term of response time or term of quality of delivered information. One of the key elements of quality is the relevance of the answers, in particular, by the taking into account of the preferences of the users in the requests.

The proposal as a whole thus defines a new approach of evaluation of the requests in preferences based on the CP-Nets. The approach suggested is founded mainly on three stage: I nterpréter accurately what the user wishes to express, these preferences and requirements was translated in a language in order to be added to the requests and to be carried out. To produce a whole of answers ordered according to the CP-Net request deduced, With this intention, we extended the operator ORD, This one allows to order the whole of the results of a relation R according to the request CP-Nets. $\succ_{Q_{CP}}$ Thus one proposed a new procedure adapted for the evaluation of the requests CP-Nets $\succ_{Q_{CP}}$ the result is a list of tuples relevant and to order for the request. Lastly, the implementation of this approach, an algorithm was proposed to manage at the same time the conditional preferences and the problem of the complexity of the calculation of the best answers under the principle *Ceteris Paribus.*

Keywords: databases, preference, preference representation, complexity, Queries to preferences, CP-Nets, Ceteris Paribus.

Remerciements

Je tiens à remercier.

Monsieur Y. DAHMANI, Maître de conférences, Université d’Ibn Khaldoun de Tiaret, qui me fait l'honneur de présider ce jury.

Monsieur M. A. CHIKH, Maître de conférences, Université d’Abou Beker Belkaid de Tlemcen, et Monsieur A. BESSAID, Maître de conférences, Université d’Abou Beker Belkaid de Tlemcen, pour l'intérêt qu'ils ont porté à ce mémoire en acceptant d'en être examinateurs.

Monsieur A. CHIKH, sans qui cette école doctorale n'aurait jamais pu commencer. Je le remercie pour le grand effort qu'a fait pour que notre école doctorale réussisse.

Monsieur A. HADJALI, Maître de conférences à l'IRISA/ENSSAT, Université Rennes 1, pour avoir encadré ce mémoire. Qu'il soit remercié pour son aide, ses conseils précieux, ses idées, ses encouragements et sa patience.

Tous ceux qui m’ont aidé de prés ou de loin a l’élaboration de ce mémoire compte tenu de leurs rependant et de l’assistance qu’ils déployèrent a mon égard sans aucune retenue.

Enfin, je remercie toute ma famille, mes amis et collègues pour leur encouragement et soutien.

Table des Matières

Table des figures

Chapitre I
Introduction générale

I.1. Contexte

Le contexte du travail est celui des requêtes à préférences adressées à une base de données relationnelle. Durant ces dernières années, de nombreux travaux ont été consacrés à la formulation et au traitement des préférences exprimées dans les requêtes des utilisateurs. Par exemple, lorsque l'utilisateur soumet une requête, il est confronté parfois à un problème de surcharge d'information[1] (En raison de l'augmentation constante du volume d'information) où il est difficile de distinguer les données pertinentes des données secondaires ou du bruit. Ceci est du au fait que les requêtes contiennent en général peu de critères et ne permettent pas de cibler très précisément le besoin de l'utilisateur.

Un des éléments clés de la qualité est la pertinence des réponses, en particulier, par la prise en compte des préférences des utilisateurs dans les requêtes. L'introduction de préférences dans les conditions de sélection vise à dépasser certaines limitations que présentent les requêtes booléennes.

Différentes propositions ont été faites pour introduire des préférences dans des requêtes. On peut distinguer deux approches générales, une approche qualitative où la préférence est spécifiée directement par des relations binaires de préférence et une autre approche quantitative où les préférences sont exprimées indirectement par le biais de fonctions qui attribuent un nombre à chaque élément et la pertinence d'un élément dépend du nombre attribué. L'approche qualitative est plus générale que l'approche quantitative car elle peut définir les relations de préférences sur des fonctions numériques si elles sont

[1] Requêtes booléennes : Tous les n-uplets de la BD, qui satisfont les conditions de la requête, sont retournés

données explicitement, tandis que toutes les relations de préférences ne peuvent pas être capturées par des fonctions numériques.

D'un autre côté, les préférences ont également fait l'objet de nombreuses études dans la communauté de l'Intelligence Artificielle. Ainsi, plusieurs approches ont été proposées pour accroître la capacité d'expression des langages de requêtes. Parmi ces approches, on s'intéresse particulièrement à celle basée sur les *CP-Nets* (*Conditional Preferences Networks*). D'une manière très concise, les *CP-Nets* sont une représentation compacte des préférences sous l'hypothèse *ceteris paribus* (toutes choses étant égales par ailleurs).

I.2. Problématique :

Dans les Systèmes Relationnels traditionnels, l'interrogation des bases de données (BD) nécessite une connaissance précise et détaillée des données et de leur organisation. L'intégration des préférences d'utilisateur[2] dans ces requêtes tente de rendre l'interrogation classique des BD plus souple pour les utilisateurs. L'intérêt majeur des requêtes à préférences est, d'une part, de pouvoir traduire fidèlement ce que l'utilisateur désire exprimer et, d'autre part, de produire un ensemble de réponses ordonnées, ce qui est particulièrement intéressant dans le cas où l'ensemble des items satisfaisant la requête est de taille importante.

Les systèmes actuels doivent être capables de traiter les requêtes à préférences. De plus, les résultats consistent en un ensemble non ordonné des n-uplets et la quantité d'information gérée par les systèmes de bases de données devient de plus en plus grande. Ces systèmes présentent des insuffisances à différents niveaux : au niveau du langage de représentation de la requête, Lors de l'interprétation et l'évaluation de la requête et de l'efficacité computationnelle du modèle.

1. Langage de représentation des requêtes à préférences

Un objectif des bases de données est d'accroître la capacité d'expression des langages de requêtes. Le traitement des préférences est l'un des éléments majeurs à considérer dans le langage de représentation de préférence. Comment représenter et comment intégrer les

[2] Ce sont des utilisateurs novices, ceux qui n'ont pas de connaissance claire sur le contenu (ex : BD accessibles par le Web).

préférences des utilisateurs dans les requêtes adressées aux bases de données ? Ce sujet a fait l'objet de nombreux travaux depuis plusieurs années [Borzsonyi et al., 2001 ; Kießling, 2002 ; Chomicki, 2003 ; Hadjali, 2007]. Une solution à ce problème consiste à représenter les préférences des utilisateurs en utilisant un langage de représentation structuré et compact. Le but est d'avoir un langage de représentation des préférences qui permet d'obtenir les réponses les plus pertinentes tout en étant ordonnées. Afin de trouver un langage de représentation compacte de préférence répondant à ces besoins, nous devons commencer par poser une question préliminaire : les préférences des utilisateurs doivent-elles être exprimées avec des préférences numériques ou ordinales ? alors que le langage de représentation de préférences doit être aussi proche que possible de la façon dont les utilisateurs "connaissent" leurs préférences et les expriment en langage naturel.

2. Sémantique et interprétation des requêtes

Une requête à préférences exprimée par un utilisateur est souvent sous forme qualitative, et on a besoin de l'analyser sémantiquement. Le résultat de cette phase est une interprétation des préférences puis l'évaluation de cette interprétation. Plusieurs modèles sémantiques existent pour l'interprétation et l'évaluation des requêtes à préférences, l'un des plus classiques étant basé sur la sémantique Totalitaire. La plupart des approches adoptent cette sémantique (*totalitaire),* c'est à dire, lors de l'évaluation des requêtes à préférences, on ne tient compte que du sous-ensemble d'attributs concernés par la requête et on ignore les autres attributs. Cependant, cette sémantique est insuffisante, et conduit parfois à des réponses controversées[3], et les types d'agrégation utilisés avec cette sémantique pour accroitre son expressivité sont trop restrictifs. Il existe une autre alternative à la sémantique totalitaire, proposée par [Boutilier et al., 2004a], c'est la sémantique Ceteris Paribus, cette sémantique est destinée à une meilleure interprétation de la notion de préférence dans les requêtes à préférences adressées à une base de données relationnelle. La sémantique Ceteris Paribus représente la façon la plus simple pour décrire les préférences des utilisateurs, le langage des CP-Nets présente un outil théorique pour l'interprétation des préférences basées sur l'hypothèse Ceteris Paribus qui permet de contourner les problèmes de la sémantique totalitaire.

[3] Pour plus de détails voir la section .4.2.2

Il serait donc intéressant d'examiner comment l'approche des *CP-Nets* peut être appliquée dans l'évaluation des requêtes à préférences sous l'hypothèse Ceteris Paribus.

3. Calcul et complexité :

Comme nous avons spécifié précédemment, l'idée principale dans les Langages de représentation des requêtes à préférences est de faciliter l'interprétation et le traitement des requêtes de l'utilisateur. La plupart des approches proposées pour le traitement des requêtes à préférences souffrent de la complexité de calcul de meilleures réponses, L'inconvénient se trouve dans la complexité des algorithmes utilisés qui nécessitent beaucoup de temps. Quel type d'algorithme utiliser et comment la contourner si elle est trop forte ?

Il serait intéressant d'aborder également le problème de la complexité du calcul des meilleures réponses sous le principe *Ceteris Paribus*.

I.3. Contribution

Notre contribution consiste à proposer une nouvelle interprétation des préférences dans les requêtes de base de données : Une approche d'évaluation des requêtes à préférences fondée sur les CP-Nets approche. L'approche est proposée afin de contourner les limitations des approches classiques.

L'approche que nous proposons se base sur trois piliers :

1. l'introduction d'un nouveau langage de requêtes exprimant les préférences qualitatives de l'utilisateur. La spécificité de ce langage concerne la prise en charge intuitive des préférences conditionnelles. Pour cela, nous exploitons l'aspect graphique du formalisme des CP-Nets pour la représentation de telles requêtes préférentielles conditionnelles. La requête est alors représentée par un CP-Net de même topologie que celui des déclarations de préférences, facilitant ainsi l'évaluation de la pertinence des données.

2. Définir et proposer une nouvelle sémantique d'interprétation pour les requêtes à préférences, sachant que la plupart des approches adoptent la sémantique *totalitaire*. Nous exploitons la logique des préférences du type Ceteris Paribus pour l'interprétation des déclarations de préférence des utilisateurs.

3. Proposer un nouveau modèle pour l'évaluation des requêtes à préférences basé sur un opérateur complexe, l'opérateur ORD [Brafman et al ; 2004] . Notre approche est fondé principalement sur trois étape : Analyse et Réécriture de la requête ; traitement de la requête CP-Nets et Génération d'un plan d'exécution. En plus, un algorithme est proposé pour calculer l'opérateur ORD, l'algorithme a pour but l'obtention d'un ensemble de tuples totalement ordonnées avec un coût optimal. En particulier, l'algorithme proposé s'affranchit d'une part des limites de la comparaison basées sur la dominance. Vu la complexité des calculs induits par cette méthode, L'algorithme se base sur les requêtes d'ordre CP-Net pour la comparaison des tuples de la BD (nœuds du CP-Nets), et la requête Q est interprétée par la sémantique Ceteris Paribus avec l'agrégation de l'union entre les différentes déclarations de préférences exprimées par l'utilisateur.

I.4. Organisation du mémoire

Ce mémoire est organisé en deux parties principales. La première partie est dédiée à la présentation des différents modèles de représentation des préférences (chapitre 2) et du formalisme des CP-Nets (chapitre 3). La seconde partie présente nos contributions. Elle est divisée en deux chapitres 4 et 5, dédiés respectivement à la présentation de notre approche d'interprétation des requêtes à préférences et d'un nouveau opérateur basé sur les CP-Nets. Le détail de cette organisation est donné comme suit :

Partie I :

Chapitre II : le deuxième chapitre est consacré à la présentation des concepts de base des requêtes à préférences. Le but de chapitre est de classifier les différents formalismes intégrant les préférences d'utilisateur dans des requêtes. L'accent est mis sur les différentes représentations implicites et explicites pour la modélisation des préférences. Les différents systèmes de requêtes intégrant les préférences sont positionnées par rapport à cette classification. Nous introduirons ensuite deux formalismes pour l'expression des requêtes à préférences, l'approche logique qui offre un cadre algébrique riche et puissant et les requêtes Skyline qui présente un modèle très prisé dans le domaine des BDs relationnel.

Chapitre III : ce chapitre sera dédié à la présentation du formalisme des CP-Nets, nous présentons des exemples illustratifs pour mieux appréhender le raisonnement du formalisme CP-Net sur lequel se base notre approche. Nous introduirons ensuite dans la partie 3 deux mécanismes d'optimisation des résultats via les CP-Nets.

Partie II :

*Chapitre IV :*Après avoir donner dans la partie I quelques éléments sur les langages de représentation des préférences, nous donnerons dans ce chapitre une description (simplifiée) des différents sémantiques d'interprétation des requêtes à préférences puis nous montrerons les travaux annexes liés à ce contexte, Nous présentons notre première contribution à la définition de notre approche d'évaluation des requêtes basées sur la sémantique Ceteris Paribus pour les BDs relationnelle

Chapitre V : L'approche proposée dans le chapitre précédent vise à étendre la sémantique Ceteris Paribus pour les BDs relationnelle, mais elle ne présente pas des mécanismes pour calculer où d'avoir les meilleures réponses à une requête donnée Q. Plus particulièrement, on propose dans ce chapitre une procédure plus générale pour l'operateur ORD qui examinera comment on peut avoir les meilleures réponses aux requêtes à préférence avec une complexité théorique plus attrayante (en termes de cout et de pertinence des réponses). Nous proposons par la suite un algorithme qui calcule et met en ouvre l'operateur ORD.

Enfin, Nous terminons par une conclusion générale et les perspectives futures de notre travail.

Chapitre II

Requêtes à Préférences

II.1. Introduction:

En raison de l'augmentation constante des bases de données, la conception et la mise en œuvre d'outils efficaces pour l'accès aux données pertinente, devient une nécessité absolue. Un des éléments clés de la pertinence est la qualité des réponses, en particulier, par la prise en compte des préférences des utilisateurs dans les requêtes. La notion de préférence joue un rôle central pour l'accès à l'information pertinente.

L'intégration des préférences d'utilisateur dans les requêtes adressées aux bases de données permet de classer les n-uplets de la base en fonction de ces préférences est d'obtenir des réponses plus pertinentes, ainsi de dépasser certaines limitations que présentent les requêtes booléennes. Par exemple, très souvent, lorsque un agent exprime une requête classique, les systèmes de base de données traditionnels délivrent des résultats massifs, c.-à-d. des réponses pléthoriques en réponse à cette requête, tous les n-uplets de la Base de Donnée (BD), qui satisfont les conditions de la requête, sont retournés, générant ainsi une surcharge informationnelle dans laquelle il est souvent difficile de distinguer les réponses pertinentes des réponses secondaires. Donc l'approche la plus simple pour remédier à cette situation est " la Reformulation de la requête " Ceci signifie le raffinement de la requête pour la rendre plus sélective. Une autre approche consiste dans la classification automatique des résultats de la requête selon leurs degrés de "pertinence" par rapport à la requête, et retourner les meilleurs. Cette approche peut être réalisée par un langage d'interrogation spécifique en prenant en compte les **préférences**[1] des utilisateurs (*langage de représentation de préférences*) .

[1] Représentation des préférences peut être explicite ou implicites

Le but de ce chapitre est de présenter les concepts de base sur les requêtes à préférences. Ce chapitre est organisé comme suit : en section 2.2, nous présentons les concepts de base sur les *requêtes à préférences*. Nous y décrivons notamment les relations de préférences qui forment la base et le fondement théorique des requêtes à préférences. La section 2.3, présente deux familles d'approches pour l'expression des préférences. Une approche implicite où chaque valeur d'attribut est associée à un score et une approche explicite où les préférences sont définies en comparant les valeurs d'attributs deux à deux. La section 2.4 détaille deux formalismes pour l'expression des requêtes. Enfin la section 2.5 conclut le chapitre.

II.2. Les requêtes à préférences :

L'intégration des préférences d'utilisateur dans les requêtes adressées aux bases de données suscite un grand engouement dans la communauté des chercheurs en BD et constitue un enjeu majeur pour l'industrie informatique. Ainsi différentes propositions ont été faites pour introduire des préférences dans des requêtes. Un des moyens étudiés est la prise en compte de préférences afin de faciliter l'accès à des informations pertinentes.

II.2.1. La notion de préférences :

Les préférences sont omniprésentes dans notre vie quotidienne, basées sur l'idée que les gens expriment leurs préférences avec le terme « je **préfère** A à B », " **préfère** ou **meilleur que** " peut être défini qualitativement [Kiessling, 2002] ou quantitativement [Agrawal et al., 2000], Les préférences peuvent être aussi complexes et couvrant des attributs multiples. Les préférences sont aussi employées pour le filtrage et l'extraction d'information pour réduire le volume de données présentées à l'utilisateur. Elles sont également employées pour capturer les profils des utilisateurs (personnaliser) et formulent des politiques pour améliorer et automatiser la prise de décision.

II.2.2. Interrogation avec des Préférences :

Adressé une requête a une base de données (BD) nécessite une connaissance précise et détaillée des données et de leur organisation. L'interrogation avec des préférences tente de rendre l'interrogation classiques des BD plus souple pour les utilisateurs. Sachant que l'interrogation classique d'une BDR est qualifiée d'interrogation

booléenne dans la mesure où l'utilisateur formule une requête, avec SQL par exemple, qui retourne un résultat ou rien du tout. Cette interrogation pose des problèmes pour certaines applications [Larsen, 1999] et [Bouaziz et al., 1997]. .

Définition 2.1 (Requête à préférences) C'est une requête qui exprime des propriétés désirées[2] sur les résultats issus du processus de recherche, c'est une extension d'une interrogation classique où l'utilisateur n'exprime pas directement son besoin avec une condition booléenne, mais au contraire avec des relations de préférence ou graduellement avec des termes imprécis.

II.2.3. Relation de préférences :

Dans les situations courantes, les préférences sont communément exprimées à base de comparaisons. C'est en particulier le cas lorsque sont utilisées des expressions comme « meilleur que », « pire que », « aussi bon que », « au moins aussi bon que ». Ces dernières ont en point commun de mettre en relation deux objets qui peuvent, a priori, être choisis. C'est pourquoi les préférences sont couramment modélisées et décrites par des relations binaires sur l'ensemble des alternatives. Par exemple, on peut décider que, pour tout couple d'alternatives (A, B), la notation (A R B) dénote le fait que choisir l'alternative A est au moins aussi préférable que choisir l'alternative B.

Une relation de préférence consiste à spécifier dans quelle mesure telle ou telle réponse est plus intéressante que telle autre, l'intérêt majeur est de trouver les meilleures réponses dominantes plutôt que toutes les réponses. On peut définir les préférences comme une relation binaire entre les n-uplets présents dans la réponse à une requête adressée à une base de données.

Définition 2.2 (Relation de préférence) [Hadjali, 2007] **:** Soit une relation de schéma $R(A_1, .., A_k)$ tel que U_i, $1 \leq i \leq k$, est le domaine de l'attribut A_i, une relation $\succ$ est une relation de préférence sur R si elle constitue un sous-ensemble de $(U_1 x \ldots x U_k) \; x \; (U_1 x \ldots x U_k)$.

Intuitivement, $\succ$ est une relation binaire entre les n-uplets de la même relation d'une BD. On dit que le n-uplet t_1 ***domine*** *(ou est* ***préférable*** *à)* le n-uplet t_2 dans le contexte de la relation $\succ$ si $t_1 \succ t_2$

[2] les utilisateurs ne cherchent pas à obtenir nécessairement toutes les réponses d'une requête, mais plutôt les meilleures réponses ou les plus "préférées".

Propriétés fréquentes des relations de préférences: Une relation de préférences est une structure mathématique utilisée fréquemment. Elle peut vérifier (ou ne pas vérifier) de nombreuses propriétés. Nous faisons ci-dessous la liste de propriétés fréquentes des relations binaires. Plus précisément, une relation (binaire) R sur A est dite :

i. Irréfléxive si et seulement si $\forall\ x\ ;\ x \not\succ x$
ii. Asymétrique si et seulement si $\forall\ x, y\ ;\ x \succ y \Rightarrow y \not\succ x$
iii. Transitive si et seulement si $\forall\ x, y, z\ ;\ (x \succ y \wedge y \succ z) \Rightarrow x \succ z$
iv. Négativement transitive si et seulement si $\forall\ x, y, z\ ;\ (x \not\succ y \wedge y \not\succ z) \Rightarrow x \not\succ z$
v. Connectivité : $\forall\ x, y\ ;\ x \succ y \vee y \succ x \vee x = y$

Par ailleurs, les propriétés ne sont pas indépendantes les unes des autres. Plus précisément, certaines sont des implications des autres ou de combinaisons des autres. En particulier, une relation Asymétrie implique Irréfléxivité, Irréfléxivité et transitivité implique asymétrie. Cependant c'est préférable de parler de ces propriétés séparément.
Parmi les relations de préférences, certaines sont d'importance dans la modélisation des préférences : les relations incomparabilité et les relations d'ordres.

Définition 2.3 (Relations incomparabilité) c'est une relation d'indifférence entre deux tuples t_1 *et* t_2*, c.-à-d.* Si ni $t_1 \succ t_2$ et ni $t_2 \succ t_1$ ne sont vérifiées, on dit que t_1 et t_2 sont *incomparables* (indifférent), la relation « t_1 est **indifférent** à t_2 » s'interprète comme suit:

$$t_1 \sim t_2$$

Il peut s'agir dans certain cas d'un refus de comparer ou d'une incapacité de comparer due à un manque de connaissances.

Définition 2.4 (Relations d'ordre) Une relation d'ordre dans un ensemble est une relation binaire dans cet ensemble qui permet de comparer ses éléments entre eux de manière cohérente. Un ensemble muni d'une relation d'ordre est un ensemble ordonné ou tout simplement un ordre.

Définition 2.5 (Relation d'ordre partial strict) Une relation est un ordre partiel strict si et seulement si elle possède les propriétés d'irréflexivité, d'asymétrie et de transitivité.
Exemple 2.2 : l'operateur | définit un ordre partiel sur $\mathbb{N}$

Définition 2.6 (Relation d'ordre faible) Une relation est un ordre faible si et seulement si elle est un ordre partiel strict et possède la propriété de Transitivité Négative.

Exemple 2.3 : l'operateur + définit un ordre faible sur $\mathbb{N}$

Définition 2.7 (Relation d'ordre total) Une relation est un ordre total si elle est un ordre partiel strict et possède la propriété de Connectivité.

Exemple 2.4 : l'operateur $\leq$ définit un ordre total sur R

Dans la plupart des applications, la relation de préférence doit satisfaire au moins les propriétés d'un **ordre partial strict,** Du point de vue utilisateur, une relation est un ordre partiel à plusieurs propriétés importantes et leur préservation est souhaitable. La restriction à un **ordre faible**, qui est une hypothèse forte, est nécessaire si la relation est représentée par une fonction de score (ou numérique) :

- $x \succ y \equiv f(x) > f(y)$ est un **ordre faible**, mais non nécessairement un **ordre total** : il peut exister deux éléments différents x_1 et x_2 tels que $f(x_1) = f(x_2)$.
- Préférences qui ne sont pas des **ordres faibles**, sont communément présentes dans les applications de base de données.
- **ordre faible** capture la situation où le domaine peut être décomposé en couches tel que les couches sont totalement ordonnées et tous les éléments d'une couche sont mutuellement incomparables.

Exemple 2.5 : Soit le schéma de la relation LIVRE (Titre, Vendeur, Prix) d'une base de données (Figure 2.1).

Cet exemple illustre comment les requêtes à préférences sont appliquées pour l'obtention du meilleur prix d'un livre donné.

Requête : " *je **préfère** un livre à un autre si et seulement si leurs ISBNs sont identique et que le prix du premier est inférieur* "

	ISBN	**Vendeur**	**Prix**
t_1	0679726691	Amazon.com	140.75 *DA*
t_2	0679726691	Alapage.com	130.50 *DA*
t_3	0062059041	Bn.com	180.80 *DA*
t_4	0374164770	Amazon.com	70.30 *DA*
t_5	0679726691	Alapage.com	210.88 *DA*

Figure 2.1 **-** Relation *livre*

On peut voir que le 2eme uplet t_2 est préférable au premier t_1 qui est a son tour préférable au troisième t_3, donc notre relation de préférence peut être formulée et interprétée comme suit : $(i . v . p) \equiv (i`.v`.p`) \wedge i = i` \quad p < p`$.

II.3. Modélisations des préférences :

Dans un cadre plus général., l'objectif est de pouvoir classer les n-uplets d'une relation à partir de plusieurs préférences élémentaires. En général., il est possible de distinguer deux familles d'approches pour l'expression des préférences. Une approche implicite où chaque valeur d'attribut est associée à un score, une valeur étant préférée à une autre si elle a obtenu un meilleur score. Une approche explicite où les préférences sont définies en comparant les valeurs d'attributs deux à deux.

II.3.1. Approche quantitative [Agrawal et al., 2000; Hristidis et al. 2001] :

Dans l'approche quantitative les préférences sont exprimées indirectement par le biais de fonctions qui attribuent un nombre à chaque élément et la pertinence d'un élément dépend du nombre attribué.

Principe :

Les préférences sont spécifiées indirectement en utilisant des fonctions de score ("scoring functions"), dites aussi fonctions d'utilité ou de préférence. La représentation par fonction de score définit un score (valeur numérique) pour chaque valeur (n-uplet) du domaine de l'attribut, ainsi un n-uplet t_1 est ***préféré*** à un autre n-uplet t_2 si et seulement si le ***score*** (t_1) est plus grand que le ***score*** (t_2)**.** Par exemple un score peut être une distance par rapport à une valeur optimale. Plus la distance est faible, plus l'élément est préféré.

Définition 2.8 (fonction de score) : *Soit une relation R avec n uplets $t_1,t_2,\ldots t_N$ et M attribues $A=\{A_1,A_2,\ldots, A_M\}$, Soit $Q=\{A_1 = q_1, A_2 = q_2, \ldots, A_M = q_M\}$ une requête sur la relation* ***R,*** *le score de chaque n-uplet* t_i *dans R, pour Q, est une fonction du score de chaque attribut individuel A_j avec la valeur cible q_j (j=1,M)*

Score (Q, t) = Score Comb ($s_1, s_2 \ldots s_M$) = $\sum_{j=1}^{M} w_j \cdot s_j$

$S_j = Score(q_j, t[A_j]) \in [0, 1]$ indique de combien la valeur $t[A_j])$ est reliée (proche) à q_j ($t[A_j]$ désigne la valeur du n-uplet t pour l'attribut A_j)

W_j est le poids indiquant l'importance relative de l'attribut A_j dans la requête Q

$t_1 \succ t_2$ Ssi Score (Q , t_1) > Score (Q , t_2)

Les fonctions de score sont généralement définies par les experts du domaine *fonctions additives et monotones* (pour que le score global soit corrélé avec les scores partiels). Deux hypothèses sont à considérer selon que les scores sont **commensurables** ou pas. Dans certaines applications, c'est l'utilisateur qui spécifie les poids (w_i) assignés aux différents attributs (*importance relative des attributs dans la requête*).

Définition 2.9 (**commensurabilité**) Dans l'hypothèse de commensurabilité, les scores d'un vecteur sont tous comparables et basés sur un référentiel d'interprétation commun. Ils peuvent donc être combinés au moyen d'une fonction d'agrégation (moyenne, moyenne pondérée, min, etc.) Pour délivrer une note globale au vecteur. En conséquence, une relation **d'ordre total** est établie entre les vecteurs de scores.

Définition 2.10 (**non commensurabilité**) Dans l'hypothèse de non commensurabilité les scores attribués aux différents attributs d'un n-uplet ne sont pas combinables. En conséquence, ils ne peuvent pas être agrégés et seul **un ordre partiel** peut être défini sur les n-uplets. En particulier, l'ordre de Pareto peut être utilisé.

Il existe de nombreux outils mathématiques performants pour manipuler les représentations numériques ou quantitatives et en particulier pour maximiser les fonctions. On peut prendre comme exemple les **TopK Queries** qui adoptent le mieux ce formalisme.

Définition 2.11 [Chengkai et al., 2005]**:**

Le principe des **"Top-K Queries"** visent à trouver *k* objets (n-uplets) avec les scores globaux les plus élevés, la réponse est un ensemble ordonné de n-uplets, contrairement à une requête simple qui retourne tous les n-uplets qui "matchent" la requête, une fonction f d'ordonnancement (Ranking) est utilisée pour classer les n-uplets correspondant aux préférences et retourne les k meilleurs. Les Top K Queries sont très utilisées dans les BD multimédias accessibles par le Web.

Top K Queries = trouver les k objets qui ont les scores globales les plus élevées

SELECT * FROM r_1, r_2, r_3 WHERE c_1 AND c_2 AND c_3	BY p_1, p_2, p_3 LIMIT k
phase (1)	phase (2) (3)

Ou r_1, r_2, r_3 sont des relations donnés ; c_1, c_2, c_3 sont des conditions booléennes (conditions simples sur une ou plusieurs attributs, ou les deux) et p_1, p_2, p_3 présente les attributs exprimant les préférences de l'utilisateur comme numériques scores.

Phase 1 *:* sélection des n-uplets vérifiant les conditions de la clause WHERE (conditions booléenne)

Phase 2 et 3 :l'ordonnancement de ces n-uplets suivant une fonction d'ordonnancement, et la délivrance d'un nombre k de résultats souhaités (*limitation des réponses*).

Exemple 2.6 : Soit la relation R de la figure suivante décrivant des personnes par leur nom, leur âge, leur poids et leur taille. Le surpoids d'une personne décrite par un n-uplet est calculé par la fonction suivante : f (t) = t.poids – (t.taille – 100).

Id	Nom	Age	Poids	Taille
#1	Mohamed	31	90	180
#2	Abdelkader	29	78	160
#3	Zakaria	22	65	170
#4	Ali	26	70	177
#5	Houari	35	85	185

Figure 2.2 - Relation *personne*

La requête « chercher les deux meilleurs réponses (k = 2) à : trouver les personnes en surpoids » utilise la fonction f comme fonction d'ordonnancement. Le résultat est donné comme suit : $f(t_2) = 18$, $f(t_1) = 10$, $f(t_4) = 7$, $f(t_5) = 0$ et $f(t_3) = -5$ où les n-uplets t_1 , t_2 , t_3 , t_4 et t_5 sont respectivement associés aux clés #1, #2, #3, #4, #5,. L'ordonnancement des n-uplets est alors : t_2, t_1, t_4, t_5 puis t_3 et seuls t_2 , t_1 sont retournés

II.3.2. Approche Qualitative :

L'approche qualitative [Lacroix et al., 1987; Kießling et al., 1994; Köstler et al., 1995; Borzsonyi et al., 2001; Govindarajan et al., 2001; Chomicki , 2002; Chomicki , 2003, Kießling , 2002; Kießling et al., 2002a; Kießling et al., 2002b] est une approche plus générale que l'approche quantitative, dans la mesure où elle offre différentes façons d'exprimer les préférences. Noter qu'il est plus facile d'exprimer les préférences qualitativement ex : « *Avec le même fabricant, je préfère la voiture la plus récente* », l'utilisation des préférences qualitatives est plus simples et plus intuitives. Même si elle n'est pas basée sur la comparaison des fonctions de score, les préférences sont exprimées explicitement et les valeurs d'attribut sont comparées les unes par rapport aux autres. Par exemple, si l'on veut indiquer ses préférences sur des toiles de maître, il est plus facile de les comparer deux à deux. Dans ce contexte, la **commensurabilité** n'a pas d'objet.

En général, les préférences utilisateurs sont considérées comme étant atomiques. Cependant, lorsque l'on considère plusieurs préférences atomiques, la relation de préférence entre deux n-uplets consiste alors à comparer les valeurs de chaque attribut les unes par rapport aux autres. Les résultats sont donnés selon le principe du Pareto-optimal ce qui interdit de prendre en considération des phénomènes de compensation entre différentes préférences.

Par ailleurs, Il est possible de définir les préférences explicites entre les n-uplets par une expression logique et de s'affranchir ainsi des préférences atomiques.

Définition 2.12 [Chomicki, 2003]**:** Soit une relation de schéma $\mathbf{R}(\mathbf{X})$ avec $\mathbf{X} = \mathbf{A_1} \times ... \times \mathbf{A_k}$. U_i, $\mathbf{1} \leq \mathbf{i} \leq \mathbf{k}$, est le domaine de l'attribut A**i**. Donc la classification des n-uplets de R est définit "qualitativement" via une relation de préférence .

Si $(t_1, t_2) \in U \times U$, $\mathbf{t_1}$ est **préférable** à $\mathbf{t_2}$ est noté $\mathbf{t_1} \succ \mathbf{t_2}$.

Reconsidérer l'exemple.2.5 précédent, les relations qualitatives induites sont :

$\{t_2 \succ t_1, t_1 \succ t_3, t_1 \sim t_4, t_1 \sim t_5, \ldots\}$

Nous présentons dans la partie qui suit, Préférences SQL qui est un modèle de requêtes à préférences, basé sur cette approche.

PreferenceSQL [Kießling, 2002] enrichit et étend SQL en lui ajoutant de nouveaux mécanismes et critère de sélection pour mieux exprimer les préférences adressées aux bases de données relationnelles.

Principe :

PreferenceSQL = **Standard SQL** + **Préférences** (en tant qu'ordre partiel strict)

PreferenceSQL est une extension du langage SQL considérant les requêtes comme composées de deux parties : l'une booléenne (partie WHERE) visant à sélectionner des n-uplets et l'autre (partie PREFERRING) spécifiant un ordonnancement des éléments sélectionnés. Dans ce cas, on obtient une meilleure flexibilité car l'utilisateur peut déterminer lui-même le sens qu'il donne à chaque fonction.

SELECT <**Selection**>
FROM <**Table**>
WHERE <**Conditions**>
PREFERRING <Soft-Conditions>
"*Représente les préférences de l'utilisateur par les constructeurs de base* **"**
GROUPING <Attribute_List>
" *Equivalent à GROUP BY* **"**
BUT ONLY <But_Only_Condition> " "
" *Seuil de qualité peut être imposé en utilisant la clause* ***but_only*** *"*
ORDER BY <Attribute_List>

A la différence des prédicats booléens de sélection obligatoires dits «**Conditions**» qui permettent de sélectionner les résultats acceptables sans l'influence des préférences, les prédicats de type «**Soft-Conditions**» sont satisfaits au mieux possible et permettent au final d'ordonner ces résultats. Les prédicats «**Soft-Conditions**» expriment des préférences sur différents attributs. D'après [Kießling, 2002], chaque préférence peut être exprimée par un opérateur. L'auteur propose la définition de préférences complexes et leur expression dans un langage formel. Les préférences peuvent être définies explicitement ou implicitement par des distances, numériques (AROUND, BETWEEN, LOWEST, HIGHEST) ou non-numériques (POS, NEG, POS/NEG, POS/POS, EXPLICIT).

PreferencesSQL se place donc dans le cadre de non commensurabilité des préférences ce qui implique un ordre partiel.

Exemple 2.7: Sur cet exemple, l'utilisateur cherche à acheter une voiture de marque Peugeot, de préférence de catégorie Peugeot 308 , mais pas Peugeot 307. Avec la même importance que la catégorie il désire que le prix soit autour de 1.400.000 Da et la puissance aussi grande que possible.

```
SELECT *
FROM Voitures
WHERE marque = 'Peugeot'
PREFERING (catégorie = 'Peugeot308' ELSE catégorie <> 'Peugeot307'
                    AND prix AROUND 1400000
                    AND HIGHEST(puissance))
```

II.3.3. Discussion :

Dans cette section, nous avons présenté deux familles d'approches pour exprimer les préférences de façon structurée et adéquate : l'approche qualitative et l'approche quantitative, chacune d'entre elles suit des modèles plus au moins différents. L'approche quantitative constitue un outil idéal pour la représentation et le raisonnement sur les préférences utilisateur. Dans cette approche les mécanismes de scores numériques commensurables ou non sont utilisés pour représenter les préférences. Un ordre total est établi sur l'ensemble des résultats lorsque les préférences sont commensurables et un ordre partiel est obtenu si les préférences sont non commensurables, le modèle le plus illustratif est les "***Top-K Queries***". Cependant, les fonctions d'utilité peuvent être très difficiles à formuler et un effort considérable est requis de l'utilisateur. Ainsi dans certains cas, il est difficile pour l'utilisateur d'exprimer ses préférences au moyen de scores. Dans l'approche qualitative les préférences sont spécifiées directement par des relations binaires de préférence, l'approche consiste à raisonner en termes d'ordres de préférence qualitatifs plutôt qu'avec des fonctions de préférences numériques. Ainsi l'approche qualitative est plus générale que l'approche quantitative car elle peut définir les relations de préférences sur des fonctions numériques si elles sont données explicitement, tandis que toutes les relations de préférences ne peuvent pas être capturées par des fonctions numériques. En effet, Pour de nombreux domaines, il est plus facile de comparer les valeurs d'attributs les unes par rapport aux autres, et dans la plupart des cas, un ordre partiel est obtenu sur les n-uplets.

II.4. Formalismes et Modélisations logiques des préférences :

Comme nous l'avons remarqué dans la partie précédente, même si cela est moins fréquent, les informations sur lesquelles se base la décision peuvent aussi être des comparaisons. C'est pourquoi nous présentons dans cette partie de chapitre deux modélisations qualitatives de la préférence. Plus précisément, après avoir exposer les hypothèses fréquemment effectuées dans ce domaine, nous décrirons le formalisme proposé par S.Borzsonyi, D. Kossmann, and K. Stocker dans [Borzsonyi et al., 2001] et celui proposé par Chomicki, J.: *Preference formulas in relational queries* dans [Chomicki, 2003].

Hypothèses fréquentes :

La connaissance dans de nombreux domaines fait intervenir des comparaisons. Souvent ce ne sont pas des comparaisons entre des alternatives à proprement parlé, mais entre des propriétés de ces alternatives. En effet, les comparaisons sont souvent effectuées afin de mettre en relation des groupes d'alternatives ayant chacun une propriété caractéristique. Aussi, depuis une cinquantaine d'années, de nombreux chercheurs en philosophie se sont proposes d'étudier la façon dont une personne pouvait exprimer ces comparaisons et en particulier les préférences. Même s'il n'existe pas de consensus sur la façon dont les préférences sont exprimées[3], il en ressort néanmoins un petit nombre de principes généraux fréquemment admis et en particulier les principes d'Expansion, de Transitivité, de Dépendance au contexte et Ceteris Paribus.

a) Expansion (ou Exclusion Mutuelle) :

Le principe le plus généralement admis est celui d'Expansion. Il indique que les préférences sont généralement exprimées par des comparaisons entre des objets mutuellement exclusifs. En particulier, si ce sont des propriétés qui sont comparées, alors celles-ci doivent être inconsistantes entre elles.

b) Transitivité :

Comme nous l'avons remarqué dans la partie précédente de la section 2.2.3, la notion d'ordre est pour beaucoup centrale aux préférences. C'est elle qui permet d'exprimer qu'un objet est meilleur qu'un autre, voire que tous les autres. Cette notion est communément modélisée avec des relations binaires transitives. Normalement appliquée

[3] Il existe plusieurs conceptions différentes de la préférence

aux alternatives, elle est aussi parfois appliquée aux propriétés. Elle indique alors que si la propriété p1 est préférée à p2 et la propriété p2 est préférée à p3, cela signifie alors aussi que la propriété p1 est préférée à p3.

c) Dépendance au contexte :

Une préférence est relative à un contexte [Hansson, 1996]. En particulier, un agent peut préférer l'objet a à l'objet b dans un certain contexte et, dans un autre, préférer l'inverse sans pour autant être qualifié d'inconsistant. En d'autre termes, cela signifie que les préférences exprimées et/ou manipulées sont généralement conditionnées par le contexte.

d) Ceteris Paribus :

Le principe Ceteris Paribus indique pour sa part que les alternatives à considérer pour donner du sens aux comparaisons, doivent vérifier les mêmes autres propriétés :

Ceteris Paribus ≡ « toutes choses égales par ailleurs »

En particulier, si la propriété p1 est dite être préférée à la propriété p2, cela signifie qu'une alternative vérifiant la propriété p1 est préférée à une alternative vérifiant la propriété p2 si elles sont « toutes choses égales par ailleurs » c'est-`a-dire si elles ont les mêmes autres propriétés.

II.4.1. Formalisme proposé par Börzsönyi « l'operateur Skyline » :

Le modèle Skyline Queries a été introduit par S. Borzsonyi, D. Kossmann et K. Stocker « *The skyline operator »* [Borzsonyi et al., 2001] . Les requêtes Skyline permettant de filtrer les résultats les plus intéressants parmi un grand ensemble des résultats. Les requêtes Skyline sont écrites dans un langage SQL étendu par l'introduction de préférences dans les critères de sélection: Tous les résultats d'une requête n'ont pas la même **importance** pour l'utilisateur. Cela permet de pallier les problèmes posés par d'autre approche quantitative ex : « les "Top-K Queries" ».

Principe :

L'opérateur Skyline est définit dans le contexte d'une base de données relationnelle interrogeable par le langage SQL. Les éléments du Skyline sont ceux qui ne sont pas dominés par d'autres éléments. Un élément domine un autre s'il est au moins aussi bon que le dominé dans toutes ses dimensions et **meilleur** dans au moins une dimension. Un élément peut être vu comme un point dans un espace à m dimensions où m

est le nombre d'attributs qu'il possède. Les requêtes Skyline sont basées sur l'idée de dominance. le principe consiste à sélectionner les meilleurs n-uplets, c'est-à-dire, ceux qui ne sont pas dominés au sens de la relation de préférence, ils sont très utilisé dans le domaine de la théorie de décisions multi critère.

Définition 2.13 (Principe de dominance): Connu sous le nom du principe d'Optimalité de Pareto en théorie de décision

- Soit un ensemble ***D*** d'objets, et ***n*** fonctions de score $S = \{s_1, s_2, .., s_n\}$
- Soient ***d1*** et ***d2*** deux éléments de **D** : d_1 domine ($\succ_s$) d_2 ssi
  - $\forall \mathbf{s} \in \mathbf{S}, \mathbf{s}(d_1) \geq_s \mathbf{s}(d2)$, et $\exists \mathbf{s} \in \mathbf{S}, \mathbf{s}(d1) >_s \mathbf{s}(d2)$.
- **d1** est au moins aussi bon que **d2** pour toutes les fonctions de score et meilleur que **d2** dans une fonction de score

Définition 2.14 (Skyline): L'ensemble des éléments maximums de D par rapport à $\succ_s$ sont appelés ***Skyline*** de D par rapport à S :

$S_S(D) = \{d \in D / \nexists d_1 \in D, d_1 \succ_s d\}$

$S_S(D)$: contient les éléments qui ne sont dominés par aucun élément sur toutes les dimensions.

La dominance est une relation transitive ($d1 \succ_s d2$ **et** $d2 \succ_s d3 \Rightarrow d1 \succ_s d3$)

II.4.1.1. Extension du SQL:

Pour son intégration, Börzsönyi [Borzsonyi et al., 2001] a proposé une extension du standards SQL par l'introduction d'une nouvelle clause optionnelle dont la syntaxe est la suivante :

SELECT FROM WHERE
***SKYLINE OF [DISTINCT] d1** [MIN | MAX | DIFF], ..., **dm** [MIN | MAX | DIFF]*
ORDER BY ..

1. ***d1***,…, ***dm*** : Représente les dimensions du Skyline (attributs de la relation); ex : prix, distance, etc.

2. ***[MIN | MAX | DIFF]*** : Les modificateur qui spécifie comment les valeurs de la dimension correspondante devrait être minimisée, maximisée ou simplement être différente respectivement.

3. ***DISTINCT*** : est un facultatif qui indique comment traiter les duplications (points identiques). Si p=q pour tous i=1…n alors p et q sont incomparables et peuvent tous les deux faire partie de Skyline si aucun DISTINCT ni indiqué, sinon p ou q est maintenu.

4. ***SKYLINE OF*** : La clause SKYLINE est traduite en langage algébrique par un nouvel opérateur logique nommé Skyline. La sémantique de cet opérateur est de sélectionner tous les n-uplets intéressants, i.e. les n-uplets qui ne sont pas dominés par d'autres n-uplets. Intuitivement, cet opérateur est exécuté après les opérateurs Scan, Join, et Group-By et avant le dernier opérateur Sort, s'il existe. Les auteurs proposent également différentes implémentations de l'opérateur Skyline [Borzsonyi et al., 2001].

Exemple 2.8 : soit la requête suivante :

SELECT *
FROM Annonces
WHERE ville **= *** Alger *****
SKYLINE OF prix **MIN,** distance_metro **MIN,**
Surface **MAX,** chambres **MAX,** propriétaire **DIFF**

Sur cet exemple, l'utilisateur cherche des annonces de logement dont le prix et la distance jusqu'à une station de métro sont minimales, la surface et le nombre de chambres sont maximales et la différence du propriétaire peut être expliquée par le fait que le client veut demander à chaque propriétaire s'il propose d'autres logements et ceci permet à l'utilisateur de consulter moins d'annonces.

II.4.1.2. Calcul et implémentations de l'operateur Skyline :

Selon [Borzsonyi et al., 2001], la clause SKYLINE OF est calculée après le SELECT... FROM... WHERE... GROUP BY... HAVING... , mais avant le reste des clauses comme ORDER BY... ou TOP N . Dans ce contexte, étant donnés deux tuples p=(p1,...,pn) et q=(q1,...,qn), p va dominer q selon la requête SKYLINE OF d1 MIN,..., dk MIN, dk+1 MAX,..., dl MAX, dl+1 DIFF,..., dm DIFF si les trois conditions sont vérifiées :

- pi<=qi pour i de 1 à k
- pi>=qi pour i de k+1 à L
- pi=qi pour i de L+1 à m

Le calcul du Skyline à n dimensions est trivial et nécessite un parcours de la table si elle est triée selon une des dimensions. Dans ce cas, il faut juste déterminer la dominance entre le tuple courant et le dernier tuple du Skyline. Si la table n'est pas triée, il est possible d'éliminer certains tuples pendant la phase de tri. L'article présente deux algorithmes de calcul du Skyline dans le cas général à *n* dimensions : sur la base de boucles imbriquées (BNL) et sur l'algorithme diviser et conquérir (D&C) :

- **L'algorithme BNL (Block-Nested-Loops):**

L'idée de l'algorithme BNL (Block-Nested-Loops, c.-à-d avec des boucles imbriquées) est la comparaison de chaque tuple avec tous les autres. Afin d'optimiser son fonctionnement, les auteurs de l'article proposent qu'à chaque itération l'algorithme ne produise pas un tuple unique, mais un ensemble de tuples. La taille de cet ensemble ou fenêtre est fixé et lors de la prise en compte d'un tuple p, trois cas sont possibles :

1. si p est dominé par un ou plusieurs tuples de la fenêtre, il est éliminé et ne va plus être pris en considération (la comparaison s'arrête au premier tuple dominant);

2. si p domine un ou plusieurs tuples de la fenêtre, ces tuples sont éliminés et ne vont plus être considérés et p est inséré dans la fenêtre;

3. si p est incomparable avec tous les tuples de la fenêtre, il y est inséré s'il y a de la place, sinon il est mis dans un fichier temporaire qui contient les tuples qui vont être traités dans l'itération suivante.

A la fin de l'itération tous les tuples de la fenêtre qui ont été comparés à tous les autres (du fichier temporaire) peuvent être retournés au résultat, les autres doivent être comparés lors de l'itération suivante. Afin de connaître les tuples à retourner, ils sont indexés à leur insertion. De cette façon tous les tuples de la fenêtre avec un numéro plus petit que celui du tuple inséré le plutôt dans le fichier temporaire peuvent être mis dans le résultat. Dans certain cas cet algorithme peut être amélioré en mettant au début de la fenêtre les tuples dominants ce qui diminue le nombre de comparaisons. Une autre variante consiste à ne garder dans la fenêtre que l'ensemble de tuples le plus dominant. La découverte d'un tel ensemble n'est pas facile et nécessite des connaissances complémentaires et des calculs de comparaison supplémentaires.

Afin d'illustrer l'utilisation de cette algorithme on considère l'exemple suivant :

Exemple 2.9 : Soit la requête suivante qui consiste à trouver les hôtels les moins chèrs du marché mais aussi les plus proches de la plage.

Requête : **SELECT** *
FROM Hôtels
WHERE ville = 'Alger'
SKYLINE OF prix **MIN,** distance **MIN;**

Pour chaque Hôtel, on à sa *distance* à la plage (axe x) et son *prix* axe (y).
Donc On cherche les meilleurs hôtels (*min* sur la distance et *min* sur le prix):

Hôtels	*Prix (DT)*	*Distance (m)*
A	45	100
B	40	200
C	42	300
D	35	400
E	50	280
F	60	150
G	55	50
H	65	250
I	72	40
J	40	500
K	68	100

Figure 2.3 - Relation *Hôtels*

Le résultat de cette requête est présenté dans la figure suivante

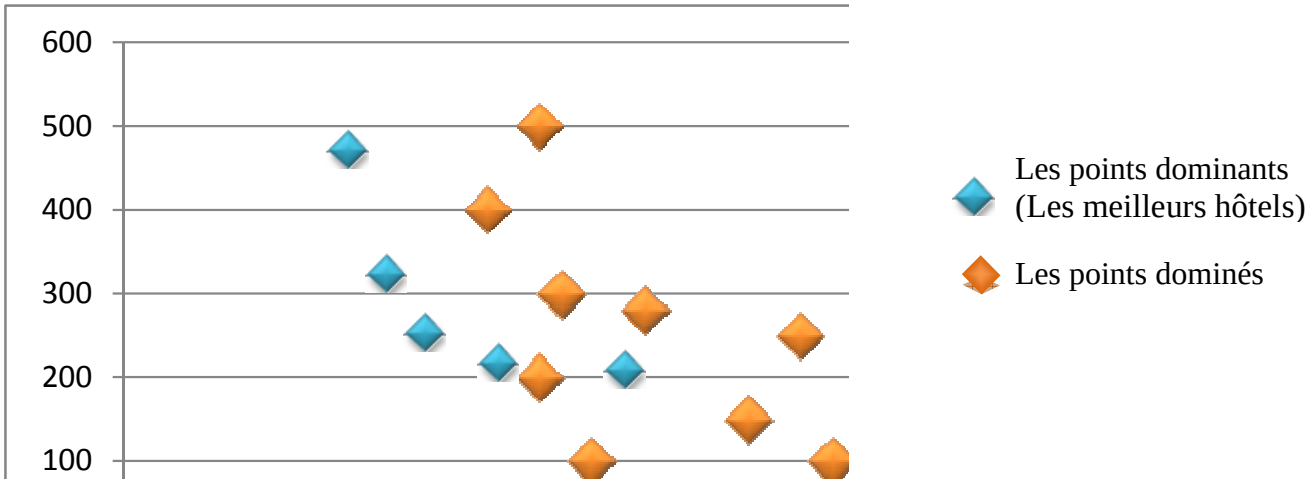


Figure 2.4 -Exemple de requête avec une clause SKYLINE OF

Le résultat est donné dans le tableau ci-dessous.

Hôtels	*Prix (DT)*	*Distance (m)*
A	45	100
B	40	200
D	**35**	**400**
G	55	50
I	72	40

***Figure 2.5*:** résultat de la requête

Le résultat est donné comme suit : Skyline = {**a, b, d, g, i**} présente les hôtels les plus intéressants pour lesquels il n y a aucun hôtel qui est meilleur par rapport aux 2 dimensions.

- **L'algorithme diviser et conquérir (D&C) :**

L'idée de l'algorithme *Diviser pour régner* est simple, basé sur le principe de la récursivité sur les données : on sépare les données en deux parties quelconques (ou plus), puis on résout les sous-problèmes (par la même fonction), pour enfin combiner les résultats. La faible complexité des algorithmes Diviser pour régner est l'un de leurs principaux intérêts. L'algorithme diviser et conquérir est basé sur l'approche suivante :

1. Calculer la valeur moyenne dp selon une des dimensions p.
2. Diviser la relation en deux parties P1 et P2 de façon à ce que P1 contient les tuples de valeur meilleure que dp dans la dimension p et P2 les autres tuples.
3. Calculer le Skyline des deux partitions en appliquant à nouveau le même procédé de partitionnement de façon récursive.

L'algorithme de découpage s'arrête lorsqu'un sous-ensemble ne contient qu'un tuple ou très peu de tuples. Et finalement le résultat est obtenu en fusionnant les sous-ensembles (il faut juste éliminer les tuples de P2 dominés par des tuples de P1 car les tuples de P1 sont meilleurs que ceux de P2 dans au moins une dimension). Cet algorithme offre de très mauvaises performances si la table ne tient pas en mémoire centrale en raison du nombre d'E/S nécessaire. Pour résoudre ce problème, l'article propose de découper la relation en m parties de façon que chacune de ces parties tienne en mémoire et appliquer l'algorithme séparément à chaque partition. Ce partitionnement peut être fait dans le premier ou le troisième pas de l'algorithme. Une autre astuce lorsque le résultat du Skyline est petit est

de charger autant de tuples que possible en mémoire centrale et d'appliquer l'algorithme dessus. Cette technique est appelée "early Skyline", elle doit être exécutée avant la phase de partitionnement et permet un premier filtrage des tuples.

II.4.2. Formalisme proposé par Jan Chomicki « Approche logique » :

L'approche logique a été introduite par Jan Chomicki " *Preference Formulas in Relational Queries*» [Chomicki, 2003] et à fait l'objet de nombreux travaux. L'approche logique est une approche formelle qualitative de formulation de préférences et leur intégration dans les langages de l'algèbre relationnelle.

Cette approche propose un cadre algébrique pour formuler des requêtes avec préférence et un opérateur algébrique « *Winnow* ». L'opérateur Winnow permet l'expression des préférences et de sélectionner les meilleurs n-uplets d'une relation données.

Définition 2.15 (Approche logique) : Tous les résultats d'une requête n'ont pas la même importance pour l'utilisateur. La recherche des meilleurs résultats selon les valeurs des attributs sélectionnés, peut être réalisée en utilisant des formules de préférence, l'opérateur Winnow [Chomicki, 2002].

Principe :

Le formalisme d'expression des préférences décrit dans [Chomicki, 2003], est basé sur les relations binaires de préférence. Les relations binaires de préférence sont définies par des formules de préférence. Une formule de préférence est une formule de premier ordre qui définit cette relation de préférence. Une relation est un ordre partiel strict si elle possède les propriétés d'irréflexivité, d'asymétrie et de transitivité. Si cette formule de préférence est définie en utilisant une formule de préférence intrinsèque[4], ses propriétés d'irréflexivité, asymétrie et transitivité peuvent être vérifiées en temps polynomial. Pour une relation de préférence P et deux éléments x et y, x <P y est interprété comme y est préféré devant x selon P.

Définition 2.16 [Chomicki, 2003] **:** Une formule logique C(t_1; t_2) qui est une formule du premièr ordre définit une relation de préférence $\succ_C$, comme suit :

$$t_1 \succ_C t_2 \text{ Ssi } C(t_1, t_2) \text{ est vraie}$$

[4] Les préférences intrinsèques portent uniquement sur les valeurs des attributs d'un élément tandis que les préférences extrinsèques prennent en compte des facteurs extérieurs comme l'origine d'un élément.

Exemple 2.10 :

Soit une relation Repas dont le schéma est : Repas (Plat, TypePlat, Boisson, TypeBoisson) et une formule de préférence C_2, définie sur cette relation comme suit :

$$(p1, tp1, b1, tb1) \succ_{C2} (p2, tp2, b2, tb2) \equiv (p1=p2 \wedge tp1='poisson' \wedge tb1 = 'jus ' \wedge tp2='Poisson' \wedge tb2='soda') \vee (p1=p2 \wedge tp1='viande' \wedge tb1='soda' \wedge tp2='viande' \wedge tb2='jus')$$

Si l'opérateur Winnow est appliqué sur la relation Repas en utilisant la formule de préférence $C_2(\omega_{C_2}$ (Repas)), alors le résultat va prendre en compte le fait que l'utilisateur préfère le jus lorsqu'il consomme du poisson et le soda en présence de la viande.

L'opérateur Skyline [Borzsonyi et al., 2001] est une mise en œuvre de **Winnow** où les préférences sont spécifiées à partir d'un jeu d'opérations prédéfinies. Car il permet de sélectionner des éléments non-dominés (et non comparables entre eux).

[Chomicki, 2003] à notamment étudier deux types de composition entre des opérateurs de préférence : composition unidimensionnelle (booléenne, fermeture transitive et composition par priorités.) et composition multidimensionnelle (Composition de Pareto et Composition Lexicographique). Les compositions booléennes décrites sont l'intersection, l'union disjointe et la somme linéaire (définies sur le même schéma R). Selon la composition booléenne, l'union ne préserve que la propriété d'irréflexivité, l'intersection les préserve toutes (irréflexivité, transitivité et asymétrie) et la différence ne préserve pas la transitivité des opérateurs de préférence.

II.4.2.1. L'opérateur « Winnow » :

L'opérateur **Winnow** est un opérateur algébrique permettant la sélection à partir d'une relation donnée, l'ensemble des n-uplets les plus préférés (*les meilleurs n-uplets*) et ceux qui ne sont pas dominés au sens de la relation de préférence $\succ_C$.

L'intérêt de Winnow se justifie lorsque la relation de préférence délivre un ordre partiel car il permet de sélectionner des éléments non-dominés (et non comparables entre eux).

Définition 2.17 (Winnow) [Chomicki, 2003] Si R est une relation de base de données et r une instance de R, alors C est une formule de préférence définissant une relation de préférence $\succ_C$ sur R, l'opérateur de **Winnow** est définit comme suit :

$$\omega_C(R)=\{t \in r \mid \nexists\, t` \in r .t` \succ_C t\}$$

Le **Winnow** retourne, seulement, tous les n-uplets dans r pour lesquels aucune alternative meilleure n'est disponible. Deux tuples qui appartiennent au résultat de l'opérateur ont soit les mêmes valeurs, soit sont indifférents entre eux (aucune dominance ne peut être établie).

Exemple 2.11 :

Soit la relation Voiture de la figure suivante (Figure 2.6) décrivant des voitures par leurs marques et leurs années.

La requête consiste à « Trouver les voiture les plus récentes avec le même fabricant »

La traduction de cette requête peut être définie par la relation de préférence $\succ_C$ en utilisant la Formule C1:

$(m, y) \succ_C (m', y') \Leftrightarrow m = m' \wedge y > y'$

n-uplet	Marque	Année
t_1	VW	2002
t_2	VW	1997
t_3	Kia	1997
t_4	Kia	2006
t_5	Peugeot	2008
t_6	Peugeot	2010
t_7	Renault	2007

Figure 2.6 - Relation **Voiture**

Ainsi le résultat du ***Winnow*** ω_{C1} (Voiture) appliqué à cette relation est le suivant :

n-uplet	Marque	Année
t_1	VW	2002
t_4	Kia	2006
t_6	Peugeot	2010
t_7	Renault	2007

Figure 2.7 - les voiture les plus préférées

1. Propriétés de l'operateur Winnow :

Dans cette section, nous allons présenter les diverses propriétés de l'opérateur Winnow, incluant *nonemptines*, *monotonicity*.

1. **Propriété de base :** Si **R** est une relation de base de données et r une instance de R, et Pour chaque relation de préférence $\succ_{C1}$ et $\succ_{C2}$ définit sur R on a :

   ω_{C1} (r) $\subseteq$ r

   ω_{C1} (ω_{C1}(r)) = ω_{C1}(r)

   $\omega_{C1 \vee C2}$(r) = ω_{C1}(r) $\cap$ ω_{C2}(r)

   ω_{Faux} (r) = r

   ω_{Vraie} (r) = ϕ

2. **Non Vacuité :** Si C définit une relation de préférence, qui est un OP stricte, et r est finie, alors pour toute instance non vide de ce schéma l'opérateur Winnow renvoie un résultat non vide ω_C (r) $\neq \phi$

3. **Containment :** Si C_1(t1, t2) $\Rightarrow$ C_2(t1, t2), alors pour toutes les instances r on a :

   ω_{C1} **(r)** $\subseteq$ ω_{C2}**(r)**.

4. **Commutativité :**

Si C1(t1, t2) $\Rightarrow$ C2(t1, t2) et $\succ_{C1}$ $\succ_{C2}$ sont des OP strictes, alors pour toutes les instances r on a : ω_{C1} **(** ω_{C2} **(r)) =** ω_{C2}**(** ω_{C}**(r)) =** ω_{C2}**(r)**

5. **Distributivité sur le produit cartésien :**

Pour chaque r1 et r2, ω_C (r_1 x r_2) = ω_C (r_1) x ω_C (r_2)

Classement avec *Winnow* : le classement est implicite dans l'approche. Il est défini en utilisant la notion de préférence itérée.

Préférence Itérée : on a la nième itération de l'opérateur Winnow ωC dans r est définie comme suit :

$$\omega_C^1(r) = \omega_C(r)$$
$$\omega_C^{n+1}(r) = \omega_C\left(r - \bigcup_{1 \le i \le n} \omega_C^i(r)\right)$$

L'opérateur Winnow peut être appliqué plusieurs fois sur une relation donnée et à chaque fois le résultat représente les meilleurs tuples non-retournés jusqu'à cette étape. Chomicki a définit aussi le weaker de l'opérateur Winnow pour lequel la propriété d'asymétrie et relâchée en gardant la transitivité et l'irréflexivité. Dans le résultat du weaker les tuples qui ne sont dominés que par des tuples qu'ils dominent (tuples similaires) sont inclus aussi.

2. Expressivité de l'operateur Winnow :

L'un des avantages majeurs de l'approche logique est qu'on peut traduire l'opérateur Winnow en algèbre relationnelle. Soit la formule de préférence C, qui est représentée par une disjonction de formules sur des variables libres correspondant aux valeurs de deux tuples t_1 et t_2 i.e. $C = D_1 \vee D_2 \vee ... \vee D_k$. Chacune des clauses D_i peut être vue comme une conjonction de trois clauses $D_i \equiv \varphi_i \wedge \psi_i \wedge \gamma_i$, φ_i (sur des variables de t_1), ψ_i (sur des variables de t_2) et γi (sur des variables communes). Dans ce cas, il existe une traduction de φ_i vers une condition de sélection Φ_i sur R et de ψ_i vers une condition de sélection Ψ_i sur q(R) où q est un retitrage de R. De même, γi peut être traduite comme une condition de jointure Γ_i entre R et q(R).

Dans ce cas on peut trouver une substitution de l'opérateur Winnow par des opérateurs de l'algèbre relationnelle pour ensuite intégrer ces opérateurs à la requête.

$$\omega_C(R) = \varrho^{-1}\left(\varrho(R) - \pi_{\varrho(R)}\left(\bigcup_{i=1}^{k}(\sigma_{\Phi_i}(R) \underset{\Gamma_i}{\bowtie} \sigma_{\Psi_i}(\varrho(R)))\right)\right)$$

q^{-1} : est l'inverse de q

$\pi_{\{A\}}$: opérateur de projection sur un ensemble d'attributs A

σ_ϕ : Opérateur de restriction sur un ensemble de critères Φ

$\bowtie$: Opérateur de jointure selon un critère Γ

$\bigcup_{i=1}^{k}$: Union d'ensembles d'éléments numérotés de 1 à k

II.4.2.2. Calcul et Evaluation de l'approche logique « Winnow » :

Dans cette section, Nous allons voir un algorithme d'évaluation de l'opérateur Winnow proposé dans [Chomicki, 2003]. L'algorithme de calcul de l'opérateur **Winnow** comporte plusieurs phases de parcours d'un ensemble S_i de tuples candidates à la ième étape. A chaque itération un seul tuple est ajouté au résultat ω_C (r) qui est initialement vide .Au début S_1 et S_2 sont initialisés par tous les tuples de la relation (S_1= S_2= r). Ensuite chaque tuple t_1 de S_1 (un seul à chaque itération) est comparé aux autres tuples de S_2.

Lors de la comparaison de deux tuples t_1 et t_2 (t_1 étant le tuple choisi pour l'itération) deux cas sont possibles :

1. Si t_2 domine t_1, alors fermer S_2 et passer au deuxième tuple de S_1
2. Sinon au contraire t_1 domine t_2, le tuple sélectionné t_1 est ajouté au résultat ω_C (r) , le traitement est répété jusqu'à ce que S_1 soit vide, à la fin des itérations ont aura

l'ensemble des n-uplets dans r pour lesquels aucune alternative meilleure n'est disponible.

Algorithme : Calcul de l'operateur Winnow (Algorithme NL) :

Entrer : r, $\succ_C$

Résultat : ω_C (r)

Début

$\omega_C(r) = \phi$

1. initialisation de S_1= r
2. **pour** chaque tuple t_1 dans S_1 **faire**
   a. initialisation de S_2= r
   b. **pour** chaque tuple t_2 dans S_2 **faire**
      Si $t_2 \succ_C t_1$ **alors** aller a 2.d
   c. sinon $\omega_C(r) = \omega_C(r) + \{ t_1 \}$
   d. fermer S_2
3. fermer S_1

Retourner ω_C (r)

Fin

Il est possible d'utiliser des contraintes d'intégrité sur le résultat de l'opérateur Winnow. Il existe deux types de contraintes locales et globales. Les premières imposent des conditions sur le contenu d'un tuple vu que les contraintes globales concernent un sous-ensemble de tuples. Les contraintes locales sont faciles à évaluer et sont facilement remplaçables par une sélection sur le résultat de l'opérateur.

II.5. Conclusion :

Nous avons présenté dans ce chapitre les concepts fondamentaux des requêtes à préférences. Le but principal d'une requête à préférences est l'intégration des préférences d'utilisateur dans les requêtes adressées aux bases de données pour permettre d'obtenir des réponses plus pertinentes. Par ailleurs, l'utilisateur doit être capable d'exprimer toutes ses préférences et exigences de façon simple. Ces exigences doivent être traduites dans un langage afin d'être ajoutées aux requêtes et exécutées. Les langages classiques comme

SQL ont une syntaxe précise et ne supportent pas la notion de préférence. Pour cette raison, ces dernières années ont vu naître des extensions du langage permettant d'exprimer des préférences. Parmi ces langages on retrouve les langages de représentation compacte des préférences.

Nous avons présenté par la suite, deux familles d'approches pour modéliser les préférences: quantitative et qualitative. Cependant, dans de nombreux domaines il est important de représenter les préférences d'un utilisateur de façon qualitative plutôt que de façon quantitative et de le faire le plus intuitivement possible. Par la suite nous avons présenté deux principaux formalismes d'interrogation avec préférences. Le formalisme des Skylines et l'approche logique, les deux approches se basent sur un ordre partiel, en conséquence, délivrent à l'utilisateur des n-uplets non dominés. Le premier formalisme, le travail de Borzsonyi et al. « *The Skyline Operator* » à fait l'objet de nombreuses recherches et d'un intérêt croissant dans la communauté des bases de données, l'opérateur Skyline propose d'étendre SQL en permettant à l'utilisateur d'avoir que les bonnes réponses à ça requête par le principe de dominance. Tandis que le travail de Chomicki, J.: « Preference formulas in relational queries » propose un cadre algébrique pour formuler les requêtes à préférences ainsi que l'opérateur algébrique « winnow ».

Nous détaillerons dans le chapitre suivant un langage de représentation graphique (son aspect graphique le rend attrayant et simple) de préférences fondé sur des préférences Ceteris Paribus.

Chapitre III

Le formalisme des CP-Nets

III.1. Introduction

Il existe nombreux formalismes et langages de représentation compacte des préférences proposés dans la littérature, on peut citer par exemple ceux qui sont basés sur la Logique des pondérations, Logique des priorités, Logique des distances, Logique des préférences du type "***Ceteris Paribus***", Logique des conditionnels, cependant les comparer est une tâche malaisée, parce que plusieurs critères pertinents existent. On s'intéresse particulièrement à celle basée sur la Logique des préférences du type "***Ceteris Paribus***" les ***CP-Nets*** (*Conditional Preferences Networks*). Cette famille de logique de préférences est construite sur le principe de l'hypothèse d'indépendance préférentielle, les alternatives sont alors décomposées en leurs caractéristiques qualitatives individuelles indépendantes et le raisonnement se faisant sur ces ordres de préférence partiels. Des travaux récents [Boutilier et al., 2004a] et [Boutilier et al., 2004b] ont exploité la structure d'indépendance préférentielle pour construire des modèles graphiques de représentation des énoncés de préférences ***Ceteris Paribus***. Les premiers travaux dans ce sens ont été entrepris par Boutilier, Brafman, Hoos et Pool dans [Boutilier et al., 1999]. Les auteurs ont proposé un graphe de représentation compacte de préférences qualitatives, le graphe CP-Net (Conditional Preference Network), qui exploite l'indépendance préférentielle conditionnelle pour la structuration des préférences utilisateur sous l'hypothèse Ceteris Paribus.

L'objectif du présent chapitre est de présenter le formalisme des CP-Nets sur lequel se base notre approche. Plus particulièrement, on présente le formalisme des CP-Nets en tant que langage de requête graphique permettant la formulation des préférences utilisateur de manière simple et intuitive.

Le chapitre est organisé comme suit, dans la section.2.1 nous présentons le formalisme CP-Nets, définitions et notations, ainsi que les concepts généraux utilisés dans la définition du modèle. La section.2.2 est dédiée à la présentation des CP-Nets par des exemples illustratifs. La section 2.3 définie la sémantique du formalisme. La section.2.4 présente le raisonnement via les C-Nets, puis nous présentons en section.2.5 le modèle des CP-Nets dans le cas ou le graphe est cyclique.

III.2. Les CP-Nets [Boutilier et al., 1999 ; Domshlak, 2002] :

Les réseaux de préférences CP-Nets (*Conditional Preference Networks*) ont été introduits en 1999 par Boutilier [Boutilier et al., 1999] comme un outil de représentation compacte des relations de préférence qualitatives. Le formalisme des CP-Nets constitue une représentation graphique très prisée pour la modélisation des préférences, il permet d'acquérir (ou éliciter) les préférences d'un utilisateur dans une représentation permettant à la fois de décrire fidèlement le modèle cognitif de l'utilisateur et de raisonner efficacement sur ce modèle. Les CP-Nets sont basés sur l'exploitation de l'indépendance préférentielle conditionnelle pour structurer les préférences sous l'hypothèse « *Ceteris Paribus* » toutes choses égales par ailleurs.

Avant d'introduire le modèle CP-Nets, nous allons présenter quelques notions et définitions du modèle.

III.2.1. Notations et définitions préliminaires :

Soit V= $\{X_1, X_1, \ldots, X_n\}$, un ensemble de variables (caractéristiques ou attributs) sur lesquelles les préférences utilisateur sont définies, étant donné un problème décisionnel fixé, et soit V' un sous ensemble de V.

La terminologie suivante est présentée selon [Boutilier et al., 1999; Boutilier et al., 2004a; Brafman et al., 2004b], on note :

- *Dom* (X_i) = {x_{i1}, x_{i2},…, x_{im}}, le domaine de valeurs de la variable X_i,
- *Asst* (V'), l'ensemble de toutes les instanciations possibles de V'. Une instanciation de V' résulte de l'affectation d'une valeur à chaque variable dans V'.
- *Asst* (V), représente l'espace de toutes les instanciations possibles sur les variables de V.
- Chaque élément dans *Asst*(V) définit une alternative.
- O = *Asst*(V) = $Dom(X_1) \times Dom(X_2) \times \ldots \times Dom(X_n)$ l'ensemble de toutes les alternatives possibles,
- Une assignation de valeur d'un sous ensemble X de V est notée x.
- La concaténation de deux assignations partielles disjointes respectivement de X et de Y est notée *xy*.
- Si $X \cup Y = V$, alors *xy* est un résultat complet (ou alternative). Il est dit « complétion » de l'assignation partielle x. comp(x) est l'ensemble des complétions de x.
- La relation de préférence, notée $\succ$, définie sur l'ensemble des alternatives O, est un pré-ordre complet sur O. D'où

$$\forall\, o, o` \in O, o \succ o` ou\ o` \succ o$$

Tel que $o \succ o`$ signifie que l'alternative *o* est au moins aussi préférée que l'alternative $o`$.

III.2.2. Formalisation des CP-nets :

Les réseaux de préférences conditionnelles CP-Net appartiennent à une famille de langages de représentation dit "graphiques". C'est une représentation qui permet de représenter des préférences conditionnelles de façon qualitative. Plus particulièrement c'est le premier formalisme graphique qui exploite l'indépendance de variables via l'hypothèse Ceteris Paribus afin de représenter des préférences de façon compacte et efficace.

Définition 3.1 (CP-Nets) Un CP-Nets sur l'ensemble des variables V = { X_1,..., X_n } est un graphe orienté acyclique de préférences conditionnelles, où les nœuds représentent des variables de V et les arcs expriment des relations de préférence entre les variables. Un lien de X à Y est interprété par "X est *parent* de Y".

L'ensemble des parents d'un nœud X est noté Pa(X). Il détermine les préférences de l'utilisateur sur les valeurs possibles. La notion fondamentale sur laquelle est construit le formalisme des réseaux de préférences conditionnelles (CP-Nets) est la notion d'indépendance préférentielle entre variables.

Toute variable X_i du graphe est instanciable dans le domaine de valeurs *Dom(Xi)* = $\{x_{i1}, x_{i2},\ldots, x_{in}\}$. Le prédécesseur d'un nœud X dans le graphe est dit son parent. On note (Pa(X)) l'ensemble des parents de X. (X ∪ Pa(X)) constitue une famille du CP-Net.

Définition 3.2 (Indépendance préférentielle) Soit V = $\{X_1,\ldots,X_n\}$ l'ensemble des variable de X. chaque variable X_i prend ses valeurs dans le domaine Dom (X_i) = $\{x_1^i,\ldots,x_i^i\}$. *Asst (V)* l'ensemble de toutes les instanciations possibles des variables de V.
Un ensemble de variables X est préférentiellement indépendant de son complément Y = V – X si et seulement si :

$$\forall\, \mathbf{x_1}, \mathbf{x_2} \in \textit{Asst}\,(X),\quad \forall\, \mathbf{y_1}, \mathbf{y_2} \in \textit{Asst}\,(Y) \quad \text{on a}$$

$$\mathbf{x_1\, y_1} \succ \mathbf{x_2\, y_1} \quad \Leftrightarrow \quad \mathbf{x_1\, y_2} \succ \mathbf{x_2\, y_2}$$

Ou $\succ$ représente une relation de préférence.

Si X est préférentiellement indépendant de son complément Y = V - X, on notera PI(X, Y). Ceci équivaut à dire que l'ordre de préférence sur les éléments x_1 et x_2 de X reste inchangé quelques soient les valeurs des éléments y_i de Y. On dit que x_1 est préférable à x_2 Ceteris Paribus (ie. Toutes choses étant égales par ailleurs).

Définition 3.3 (Indépendance préférentielle conditionnelle) Soit X, Y et Z trois ensembles non vides qui partitionnent l'ensemble V. X est conditionnellement préférentiellement indépendant de Y étant donnée une instanciation z ∈ Z si et seulement si :

$$\forall\, \boldsymbol{x_1, x_2} \in \textit{Asst (X)},\ \forall\ \boldsymbol{y_1, y_2} \in \textit{Asst (Y)}\ ,\ \forall\, \boldsymbol{z} \in D(Z)\ \textit{on a :}$$

$$\boldsymbol{x_1\, y_1\, z} \succ \boldsymbol{x_2\, y_1\, z} \quad \Leftrightarrow \quad \boldsymbol{x_1\, y_2\, z} \succ \boldsymbol{x_2\, y_2\, z}$$

Pour une valeur de Z fixée, la relation de préférence sur les instanciations de X est la même quelles que soient les valeurs des instanciations de Y. alors X est conditionnellement préférentiellement indépendant de Y étant donné Z (on note CPI (Y, Z, X)).

A chaque variable X du CP-Net, on associe une **table de préférences conditionnelles** (TPC(X)) spécifiant un ordre de préférence total $\succ$ sur les valeurs xi de X étant donné chaque instance de ses parents p$\in$ D (Pa(X)).

- Pour les nœuds racines X_i, la table de préférence conditionnelle, noté TPC (X_i) , exprime la préférence entre X_i et sa négation $\neg X_i$, toutes choses étant égales par ailleurs.

- Pour les autres nœuds X_j, TPC (X_j) décrit les préférences entre X_j et $\neg X_j$, toutes choses étant égales par ailleurs, étant donné une instanciation des parents de X_j .

La structure des énoncés d'indépendance préférentielle conditionnelle ainsi obtenus constitue le graphe CP-Net.

Définition 3.4 (CP-Nets) Un CP-Net est un couple G = (V, E), où V est un ensemble de nœuds V = {X_1, X_2, X_3,… X_n} qui définissent les variables de préférence et E l'ensemble des tables (une par nœud). Le graphe représente l'ensemble des dépendances entre les variables : les parents d'une variable sont les variables qui influencent les préférences sur les valeurs de cette variable. Chacune de ces tables (TPC (Xi)) décrit un ordre total sur les valeurs de la variable Xi pour chaque valuation des variables des parents de Xi (Pa (Xi)).

Un CP-Net N sur le graphe G est dit complet si chacune de ses tables est complète. Il est dit acyclique si le graphe G est acyclique, et arborescent si G est une forêt, c'est-à-dire si chaque variable à au plus un arc entrant/parent dans G, et G ne contient pas de cycle.

III.2.3. Exemples sur les CP-Nets

Nous illustrons le formalisme des CP-net et certaines de ses conséquences avec les exemples suivants.

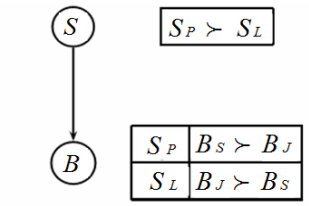
Exemple 3.1 : Soit le CP-net présenté sur la Figure 3.1 qui exprime mes préférences sur le menu d'un dîner. Le CP-net est composé de deux variables, S et B qui correspondent respectivement à la soupe et a la Boisson.

« *Je **préfère strictement** manger une soupe de poisson (S_P) plutôt qu'une soupe de légumes (S_L), tandis que mes **préférences** sur la Boisson Soda (B_S) ou jus (B_J) **dépendent** de la soupe que je mangerai. En effet je **préfère** du jus avec une soupe de légumes, mais avec une soupe de poisson je **préfère** une soda* ».

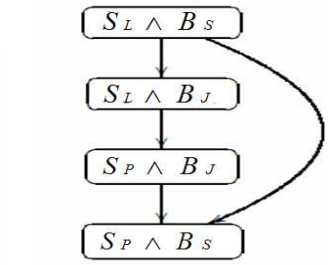
On a donc :- *Dom(S) = {S_P, S_L}. qui correspond au domaine de la soupe*

- *Dom(B) = {B_S, B_J}. qui correspond au domaine de la Boisson*

Le CP-Net qui encode mes préférences sur les variables S et B est ainsi défini à travers le graphe illustré en Figure 3.1. (a)



(a)



(b)

Figure 3.1 - (a) le CP-Nets « mon diner » - (b) Graphe de préférences induit par le CP-net

Nous pouvons donc ordonner totalement les états possibles (du plus préféré au moins préféré) : $(S_L \wedge B_S) \succ (S_L \wedge B_J) \succ (S_P \wedge B_J) \succ (S_P \wedge B_S)$

➢ **Conséquence de la définition :**

Le CP-net de l'exemple 3.1 induit une relation binaire sur les assignations complètes des variables du problème (i.e. les alternatives). Cette dernière peut être illustrée par le graphe (voir Figure.1.3 (b), Figure.3.2 (b),) dont les nœuds représentent les alternatives possibles et dont les arcs représentent les préférences entre ces alternatives : il existe un arc (orienté) du nœud o au nœud o`si et seulement si les assignations en $o$ et $o$` différent seulement de la valeur de la variable X, et si, étant données les valeurs données aux parents de X (par o et o` ), la valeur assignée par $o$`à X est préférée à la valeur assignée par o à X.

L'intérêt du CP-net est de permettre de représenter très succinctement cette relation binaire sur l'ensemble des alternatives. Pour s'en convaincre, considérons le graphe de préférences sur les alternatives possibles définies par les informations de l'exemple suivant (voir la Figure 3.2). Ce dernier est bien plus grand et complexe à appréhender.

Exemple 3.2 : « tenue de soirée »

Afin d'illustrer cette définition, considérons l'exemple suivant qui exprime les préférences d'une personne au sujet des différents vêtements possibles à mettre pour sortir. Ces préférences portent sur les quatre variables (binaires pour plus de simplicité) V , P, C et S indiquant respectivement les couleurs de la veste, du pantalon, de la chemise et des sandales à mettre.

> *« L'agent **préfère** de manière **inconditionnelle** la couleur noire à la couleur blanche pour la **veste et** le **pantalon**, alors que sa préférence entre **chemise** rouge et **chemise** blanche est **conditionnée** par la combinaison des couleurs de la **veste** et du **pantalon**. Si les deux sont de la même couleur alors il **préfère** une **chemise** rouge. Par contre, si la **veste** et le **pantalon** sont de couleurs différentes alors il **préfère** une **chemise** blanche. Enfin ,si il a une **chemise rouge**, il préfère mettre des **sandales rouges**. Dans le cas contraire, il **préfère** mettre des **sandales noires**.*

Chaque variable a un domaine fini :

- *Dom(V) = { V_N, V_B} qui correspond au domaine de la **veste***
- *Dom(P) = {P_N, P_R}. qui correspond au domaine du **pantalon***
- *Dom(C) = {C_R, C_B} qui correspond au domaine de la **chemise***
- *Dom(S) = {S_R, S_B} qui correspond au domaine des **sandales***

Ces préférences peuvent se traduire sous la forme d'un CP-net impliquant des variables binaires, présenté dans la Figure 3.2. La relation de préférence induite par ce CP-net est présentée dans la Figure 3.3.

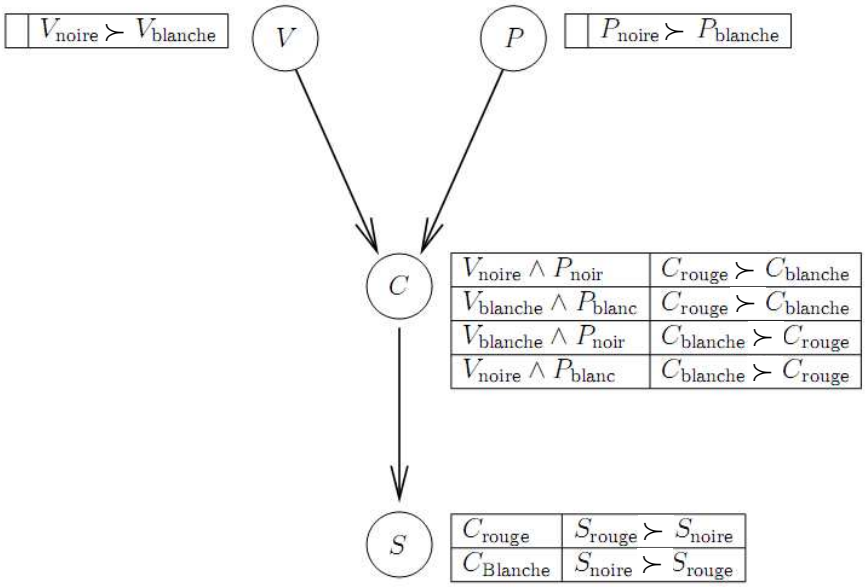


***Figure 3.2** : Exemple de CP-Nets « tenue de soirée » .*

La relation de préférence capturée par le CP-Nets introduit un ordre de préférence partiel sur l'ensemble des assignations aux variables du CP-Nets. Cet ordre partiel peut être représenté par un graphe de préférences orienté et acyclique. Les nœuds du graphe sont des alternatives (i.e. des assignations à toutes les variables) du CP-Nets. Une relation du nœud $x_i = V_b \wedge P_b \wedge C_b \wedge S_r$ vers le nœud $x_j = V_b \wedge P_b \wedge C_b \wedge S_n$ signifie que x_j est une alternative plus préférable à x_i .

Par convention, le graphe des préférences induit est ainsi ordonné par ordre de préférences qualitatif décroissant :

1. Dans la Figure 3.3 le sommet du graphe de préférences représente l'alternative la ***moins préférable*** (The WORST) ,

2. la feuille du graphe représente l'alternative la ***plus préférable*** (The BEST).

Le graphe des préférences induit par le CP-Net de la Figure 3.2 est donné en Figure 3.3 Dans ce cas, le meilleur choix de l'utilisateur (i.e. l'alternative correspondant à sa plus haute préférence 1.) Est $V_n \wedge P_n \wedge C_r \wedge S_r$, alors que le résultat correspondant à l'alternative la moins préférable est $V_b \wedge P_b \wedge C_b \wedge S_r$. Le passage d'une alternative à une autre ce fait par un changement de sa valeur initiale à sa valeur immédiatement plus préférable (*voir section 5 pour plus de détail*).

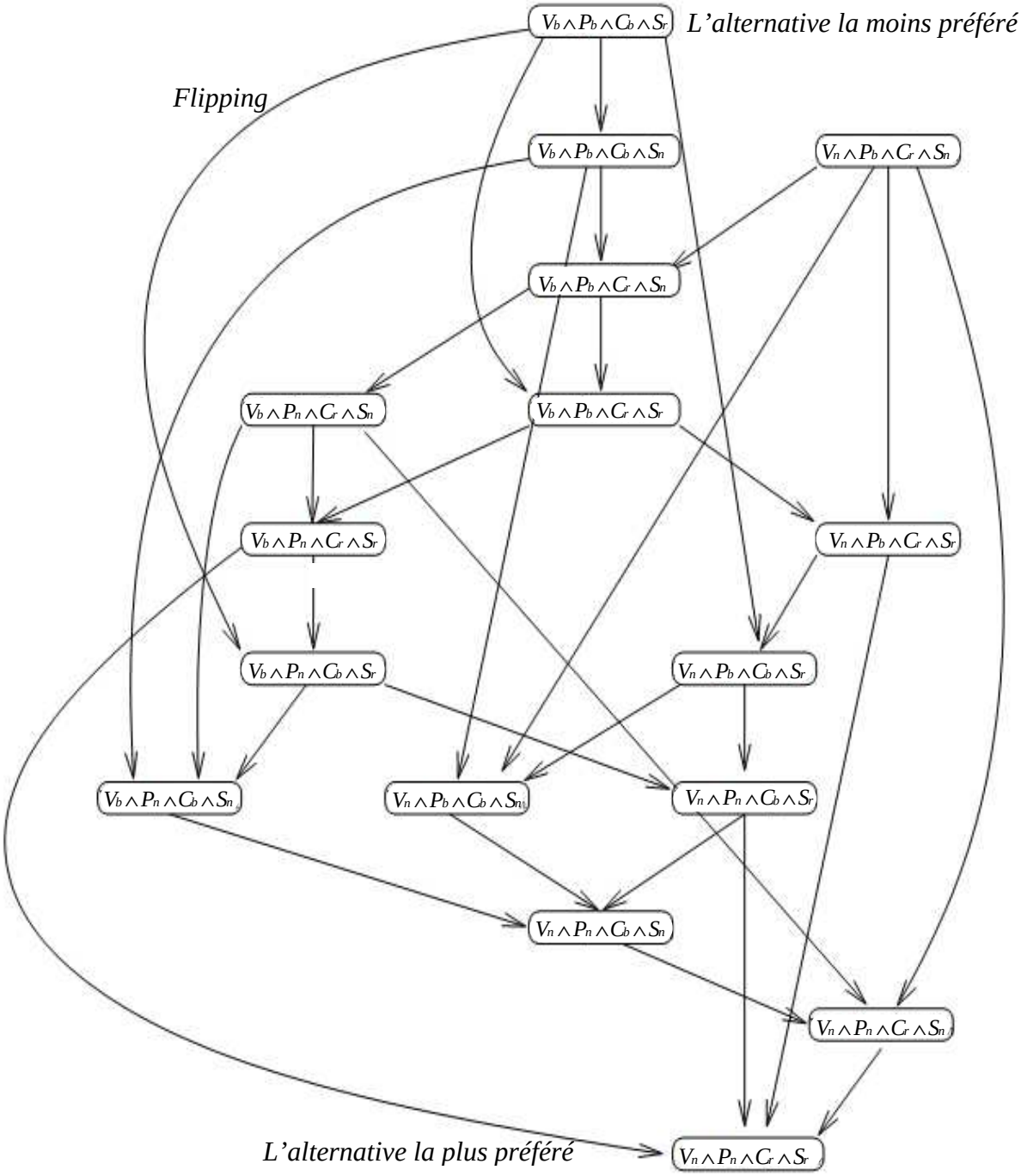


***Figure 3.3** : Graphe de préférences induit par le CP-net précédant "Tenue de Soirée "*

La signification des arcs est la même que dans la figure 3.2 : « est moins préféré à ».

III.3. La Sémantique des CP–Nets :

La sémantique des CP-nets est simple, définie en termes d'ensemble *d'ordre de préférence* (*ou relation de préférences)* qui sont consistant avec l'ensemble de contraintes imposé par les tables TPCs. La sémantique des CP-nets été principalement étudiée dans [Domshlak, 2002], [Boutilier et al., 2004a] ou encore [Boutilier et al., 2004b].

Informellement, un CP-net N est satisfait par une relation de préférence $\succ$ si $\succ$ satisfait chacune des préférences conditionnelles exprimées dans les TPCs de *N* sous l'interprétation *Ceteris Paribus*.

Définition 3.5 (Sémantique des CP-nets) Soit *N* un CP-Nets sur l'ensemble des variables V , soit X $\in$ V une variable, U $\subset$ V l'ensembles des parents de X dans *N* et Y = V – (U $\cup$ {X}).Soit $\succ_u$ une relation d'ordre sur D(X) dictée par la table de préférences conditionnels TPC(X) pour l'instance u $\in$ *D*(U) parents de X et finalement soit $\succ$ une relation de ***préférence*** sur *D*(V) = { *Dom* (X_1) $\times$ *Dom* (X_2) $\times \ldots \times$ *Dom* (X_n)}.

1. Une relation de préférence $\succ$ satisfait $\succ_u$ si et seulement si $yux_i \succ yux_j$, pour tout y $\in$ D (Y), a chaque fois que $x_i \succ_u x_j$.
2. Une relation de préférence $\succ$ satisfait TPC(X) si et seulement si $\succ$ satisfait $\succ_u$ pour chaque u $\in$ D (U).
3. Une relation de préférence $\succ$ satisfait le CP-Nets N si et seulement si elle satisfait la TPC (X) pour chaque variable X.

Définition 3.6 (CP-net est satisfiable) Un CP-net est satisfiable si et seulement s'il y a au moins une relation $\succ$ qui le satisfait.

Cette définition spécifie comment les tables de préférences conditionnelles doivent être interprétées dans le graphe : pour chaque variable x, le CP-net spécifie une relation de préférence portant sur son domaine, et dépendant de la valeur des parents de x dans le graphe. Une relation de préférence sur x et Pa(x) est « compatible » avec le CP-net si elle respecte la relation de préférences sur Dx pour toute instanciation des parents de x.

De manière générale, il peut exister plusieurs ordres qui satisfont un CP-net donné. La structure de préférence induite par un CP-net est donc définie comme l'intersection de tous ces ordres.

Observer que les CP-nets dans les exemples précédents (3.1 et 3.2) sont satisfiables puisqu'ils indiquent des ordres partiels au-dessus des résultats (c.-à-d., les graphiques des relations indiquées de préférence sont acycliques).

Le théorème 3.1 prouve que ce n'est pas une coïncidence.

Théorème 3.1 [Domshlak, 2002] **:** *« Chaque CP-net acyclique est satisfiable. »*

Généralement, la plupart des CP-Nets acycliques satisfiables sont satisfaits par plus d'un ordre de préférences.

Exemple 3.3: Etant donné un ensemble de 3 variables V= {A, B, C} binaires définies Par Dom(A)= {a_1, a_2}, Dom(B)= {b_1, b_2} et Dom(C)= {c_1, c_2}. Mes préférences sur les valeurs de ces trois attributs sont définies comme suit :

1. je préfère inconditionnellement a_1 à a_2 (i.e. $a_1 \succ a_2$),
2. mes préférences sur les valeurs de *B et C* dépendent des valeurs prises par A. Ainsi, si *A* prend la valeur a_1, je préfère b_1 à b_2 et c_1 à c_2, sinon je préfère b_2 à b_1 et c_2 à c_1.

La figure suivante présente les préférences induites

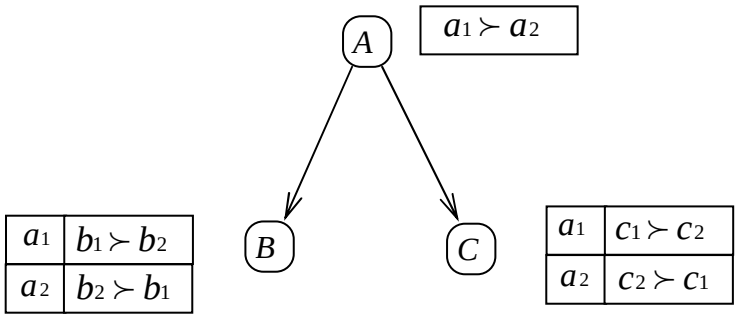


***Figure 3.4** : Exemple de CP-Nets*

D'après cet exemple, il existe quatre ordres de préférences qui satisfont le CP-Net :

$a_1b_1c_1 \succ \underbrace{a_1b_1c_2 \succ a_1b_2c_1}\succ a_1b_2c_2 \succ a_2b_2c_2 \succ a_2b_1c_2 \succ a_2b_2c_1 \succ a_2b_1c_1$

$a_1b_1c_1 \succ \underbrace{a_1b_2c_1 \succ a_1b_2c_2}\succ a_1b_2c_2 \succ a_2b_2c_2 \succ a_2b_1c_2 \succ a_2b_2c_1 \succ a_2b_1c_1$

$a_1b_1c_1 \succ a_1b_1c_2 \succ a_1b_2c_1 \succ a_1b_2c_2 \succ a_2b_2c_2 \succ \underbrace{a_2b_2c_1 \succ a_2b_1c_2}\succ a_2b_1c_1$

$a_1b_1c_1 \succ a_1b_1c_2 \succ a_1b_2c_1 \succ a_1b_2c_2 \succ a_2b_2c_2 \succ \underbrace{a_2b_1c_2 \succ a_2b_2c_1}\succ a_2b_1c_1$

Définition 3.7 (Structure de préférence induite par un CP-net) Soit N un CP-Nets sur l'ensemble des variables V, et o, o` ∈ D(V) deux alternative. N **infère** (entraine) $o \succ o`$, noté $N \models o \succ o`$, ssi $o \succ o`$ est vraie dans chaque relation de préférence compatible avec N (qui satisfait N). On dit alors que $o \succ o`$ est une ***conséquence*** de N.

Ainsi, dans le cas du CP-Nets précédent (exemple 3.3) on a :

$N \models a_1\, b_2\, c_2 \succ a_2\, b_2\, c_2$

Mais $N \not\models a_1\, b_1\, c_2 \succ a_1\, b_2\, c_1$ car il existe un ordre de préférences dans lequel

$N \models a_1\, b_1\, c_2 \succ a_1\, b_2\, c_1$

La déduction préférentielle pour un CP-Net est transitive.

Lemme 3.1 ([Boutilier et al., 2004a]). La conséquence préférentielle en ce qui concerne un CP-net est transitive. Donc :

Si $N \models o \succ o`$ et $N \models o` \succ o``$ alors $N \models o \succ o``$

- **Importance des parents sur les enfants :**

La sémantique Ceteris Paribus du CP-Net implique que les préférences sur les parents ont une priorité supérieure à celles de leurs descendants. Ainsi par exemple, dans le CP-Net de l'exemple précédent, on a $a_1\, b_2\, c_2 \succ a_2\, b_1\, c_1$ **:** la plus préférable valeur de A combinée avec les valeurs les moins préférables de B et C, donne une alternative plus préférable que celle combinant la valeur la moins préférable de A avec les valeurs les plus préférables pour B et C étant donnée cette valeur de A.

III.4. Le raisonnement par les CP Nets

Le but principal de n'importe quel modèle de représentation de préférence est de supporter les diverses requêtes à préférences du décideur et de permettre de comparer des alternatives entre elles et de déterminer la (ou une des) meilleure(s).

Parmi toutes ces requêtes on distingue deux types de requêtes :

- *Les requêtes d'Optimisation de résultats* : déterminant les meilleurs résultats (ou un des résultats non dominés, si les meilleurs résultats ne sont pas uniques).
- *Les requêtes de comparaison* : la comparaison préférentielle entre une paire de résultats (ou entre deux alternatives).

III.4.1. Optimisation des résultats :

Concerne la recherche de la meilleure alternative possible. Soient *N* un CP-net acyclique, chercher le résultat optimal du CP-net N consiste intuitivement à parcourir le graphe des préférences du sommet vers les feuilles (c'est-à-dire des parents aux descendants), en instanciant chaque variable parcourue à sa plus préférable valeur étant données les instanciations de ses parents. Cette procédure est appelée recherche avant (*forward sweep*).

Définition 3.8 (Ensemble des résultats optimaux) Soit N un CP-net sur les variables V. D(V) dénote l'ensemble de toutes les instanciations de V. On appelle *ensemble des résultats optimaux* de N l'ensemble d'états *O* tel que $\forall o \in O, N \models o \succ o'$ pour tout $o' \in$ D(V) -W

Remarque : Même si le CP-Net ne détermine pas un ordre de préférence unique, il détermine une meilleure **alternative** unique.

De façon plus générale, étant donnée une contrainte sur quelques variables $Z \subseteq V$, sous forme d'une instanciation donnée z de Z, déterminer l'alternative la plus préférable consiste à parcourir le graphe des préférences de haut en bas, en assignant à chaque variable $X \neq Z$, sa plus préférable valeur étant donnée l'instanciation de ses parents.

Lemme 3.2 (procédure de recherche en avant) [Boutilier et al., 1999] Soit N un CP-net acyclique sur un ensemble de variables V. La procédure de recherche en avant (forward sweep procedure) construit le résultat optimal dans D(V).

Exemple 3.4 : Considérons à nouveau l'exemple 3.2 de tenue de soirée dans lequel un agent doit exprimer ses préférences sur quatre « Variables » : la Veste (V), le Pantalon (P), la Chemise (C) et celui des sandales (S).

Pour déterminer l'élément préféré du décideur, on peut :

1. Choisir la meilleure alternative pour les nœuds sans parents

2 Pour chaque nœud dont les parents ont été instancies, choisir la meilleure alternative

Le graphe des préférences induit par le CP-Net de l'exemple 3.2 est redonné en figure 3.5. Dans ce cas, le meilleur choix de l'utilisateur (i.e. l'alternative correspondant à sa plus haute préférence) est $V_n \wedge P_n \wedge C_r \wedge S_r$ alors que le résultat correspondant à l'alternative la moins préférable est $V_b \wedge P_b \wedge C_b \wedge S_r$.

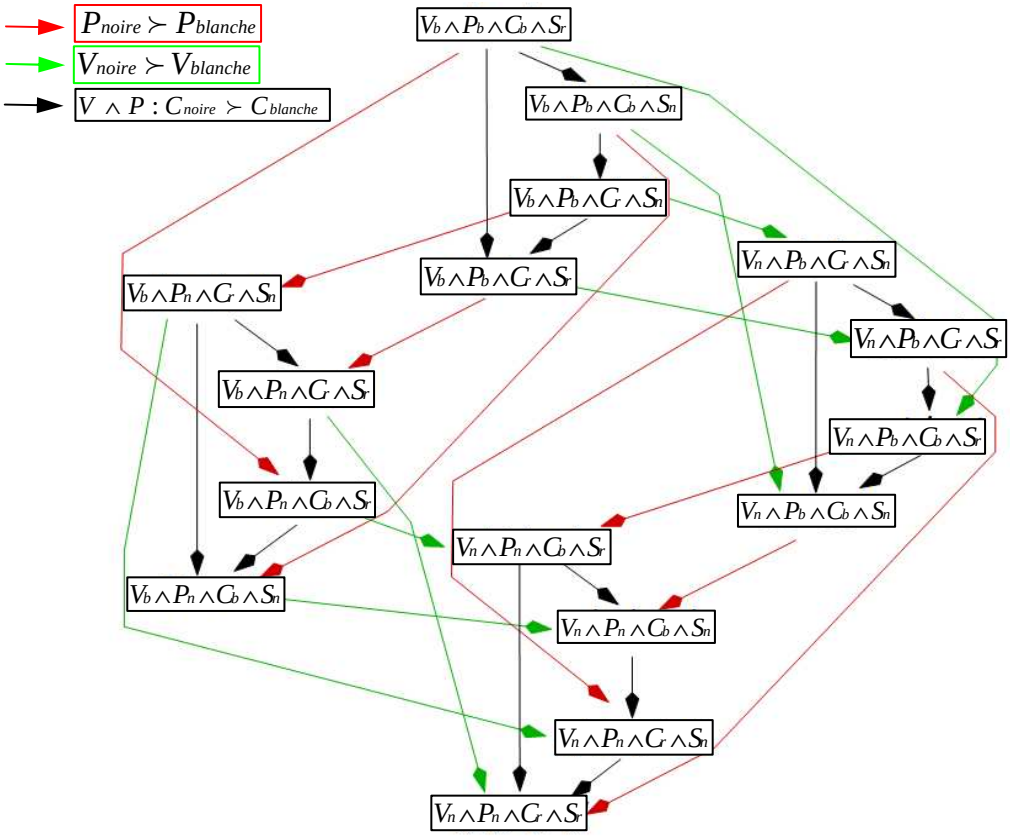


Figure 3.5. Optimisation avec les CP-nets

III.4.2. Comparaisons des résultats:

Le problème crucial en représentation des préférences est de pouvoir répondre à toute requête sur la comparaison de deux alternatives quelconques. Etant données o et $o`$ deux alternatives, déterminer si $o \succ o`$?

On distingue classiquement deux types de requêtes pour comparer des alternatives sur la base de préférences codées par un CP-net:

- *Les requêtes de dominance.*
- *Les requêtes d'ordre.*

III.4.2.1. Les requêtes de dominance :

Le problème de dominance dans un CP-Net N, d'une alternative o sur une alternative $o`$, peut être posé comme suit: $N \models o \succ o`$? Et o doit être ordonné avant $o`$

Définition 3.9 (Une requête de dominance) Soit *N* un CP-net et deux alternatives o *et* $o`$. *Une requête de dominance* consiste à vérifier si $N \models o \succ o`$. Si cette relation est vraie alors o est préférée à $o`$, et on dit que $o$ ***domine*** $o`$ par rapport à *N*.

La dominance dans les CP-Nets a été largement étudié dans [Boutilier et al.,2004a] et [Boutilier et al., 2004b], et il a été montré qu'il existe une relation direct entre l'existence d'une séquence de ***flipping*** entre une paire d'alternative et la relation de dominance entre elles.

Afin de comparer des alternatives entre elles, la notion de changement élémentaire (dénommé Flip) est introduite. Il existe un changement élémentaire (Flip) entre deux alternatives si elles ne différent que par la valeur d'une des variables (i.e si on passe de l'une à l'autre simplement en changeant la valeur d'une variable). Ce changement est dit aggravant ou détériorant (worsening flip) si l'alternative ainsi obtenue est moins préférée que l'alternative initiale.

Définition.3.10 (**séquence améliorante de flips ou séquence détériorante de flips**) Soient N un CP-net et deux alternatives o et $o`$, une séquence améliorante de flips (séquence détériorante de flips) de $o$ vers $o`$ fourni la preuve que o est plus préférable (moins préférable) à $o`$dans tout les ordres qui satisfont le CP-Nets N.

Exemple 3.4 – suite de l'exemple 3.2 :

A titre d'exemple. Est-ce que $V_n \wedge P_n \wedge C_b \wedge S_r \succ V_b \wedge P_b \wedge C_b \wedge S_n$?

Les liens internes sont construits de proche en proche en flippant une variable à la fois (en commençant par les nœuds les plus internes du CP-Net) de sa valeur actuelle à sa valeur immédiatement plus préférable étant donnée la valeur de ses parents. Ainsi de $V_b \wedge P_b \wedge C_b \wedge S_n$, en flippant la variable *C* de sa valeur actuelle C_b à sa valeur immédiatement plus préférable étant donnée $V_b \wedge P_b$, soit C_r, on obtient $V_b \wedge P_b \wedge C_r \wedge S_n$. En flippant la valeur *P* de sa valeur P_b à sa valeur immédiatement plus préférable, soit donc P_n, on obtient $V_b \wedge P_n \wedge C_r \wedge S_n$. L'alternative immédiatement plus préférable à $V_b \wedge P_n \wedge C_r \wedge S_n$

s'obtient alors en flippant la valeur de *S*, de S_n à S_r, ce qui donne l'alternative $V_b \wedge P_n \wedge C_r \wedge S_r$. A partir de $V_b \wedge P_n \wedge C_r \wedge S_r$, on construit l'alternative immédiatement plus préférable en flippant *C*, de sa valeur courante C_r à sa valeur immédiatement plus préférable C_b, ce qui donne l'alternative $V_b \wedge P_n \wedge C_b \wedge S_r$ en procédant comme précédemment, on construit de proche en proche les alternatives immédiatement plus préférables en flippant une à une les variables à leurs plus préférables valeurs étant donnée la valeur de leur parent ce qui nous mène à $V_n \wedge P_n \wedge C_b \wedge S_r$.

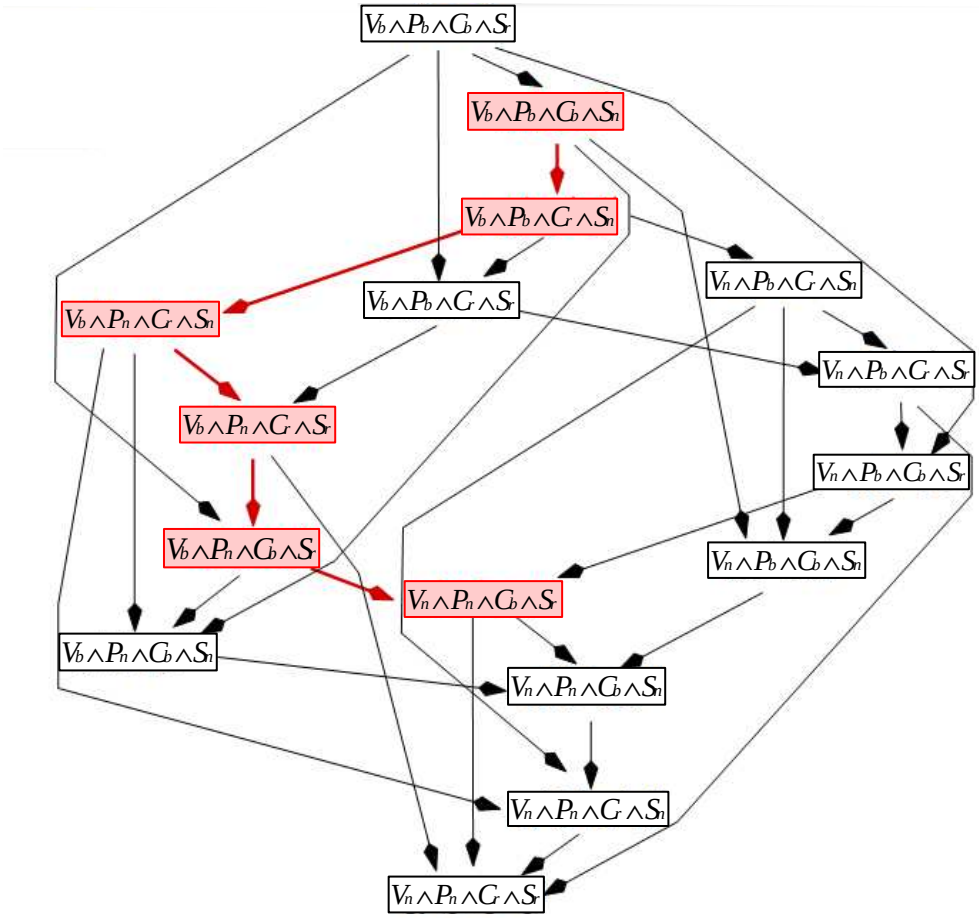


Figure 3.6 séquence améliorante de flips.

- **Complexité des requêtes de dominance :**

Dans cette section, nous donnons quelques résultats sur la complexité du test de la dominance à l'égard des CP-Nets, montrant un lien direct entre la structure du CP-graphe et le pire des cas. D'un point de vue computationnel, la complexité des problèmes de comparaison et de dominance pour les CP-Nets est synthétisée sur le tableau suivant (voir [Domshlak, 2002] pour plus de détails).

Degré du graphe	Test de dominance
Direct Tree	$O(n^2)$
Polytree ($indegree \leq K$)	$O(n^{2k}n^{2k+3})$
Polytree	NP-complete
Singly connected ($indegree \leq K$)	NP-complete
DAG	NP-complete
General case	PSPACE-complete

Figure 3.7 Tableau de la complexité du test de dominance

On voit que la complexité computationnelle du test de dominance varie énormément selon la nature du CP-Nets, comme la taille d'un CP-Nets augmente de manière exponentielle avec le degré de son graphe, la complexité peut devenir prohibitive, et ce type de problème a en général une complexité supérieure à celle des problèmes de NP, et ne peut pas être traite par des outils restreints à NP. C'est pourquoi la plupart des modèles exploités en pratique sont fondés sur des graphes de petit degré [Boutilier et al., 2004]. nombreux algorithmes existent pour résoudre une grande variété de problèmes, de complexités variées.

III.4.2.2. Les requêtes d'ordre :

Le problème d'ordonnance dans un CP-Net N, d'une alternative O sur une alternative $O`$, peut être posé comme suit: $N \not\models O` \succ O$? Et O Peut être ordonné avant $O`$

Contrairement aux requêtes de dominance, les requêtes d'ordre permettent de construire un **ordre** complet compatible avec le CP-net de tout ensemble d'alternatives.

Définition 3.11 (requêtes d'ordre) : Soient *N* un CP-net, O *et* $O`$ deux alternatives. ***Une requête d'ordre*** consiste à vérifier si $N \not\models O` \succ O$. Si cette relation est vraie alors il existe un ordre de préférence compatible avec *N* dans lequel nous avons $O \succ O`$, i.e., cette relation de préférence est compatible avec la connaissance exprimée par *N* pour préférer $O$ *à* $O`$. Dans un tel cas, O est dite "***ordonnable***" de manière cohérente par rapport à $O`$ étant donné *N*.

Même si les requêtes d'ordre sont plus faibles que les requêtes de dominances elles sont très utiles dans nombreux domaines et elles ont l'avantage d'une complexité de calcul moindre.

Exemple 3.5 Soient A, B et C trois variables binaires. La Figure 3.8 suivante présente le CP-Nets sur ces variables. Les préférences associées à ce CP-Nets sont :

$aBC \succ \neg aBC, \forall B \in \{b, \neg b\}, \forall C \in \{c, \neg c\},$
$abC \succ a\neg bC, \forall C \in \{c, \neg c\},$
$\neg a\neg bC \succ \neg abC, \forall C \in \{c, \neg c\},$
$Abc \succ Ab\neg c, \forall A \in \{a, \neg a\}, et$
$A\neg b\neg c \succ A\neg bc, \forall A \in \{a, \neg a\}.$

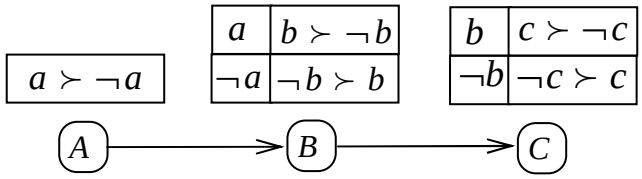


Figure 3.8 Le CP-net associé aux préférences de l'exemple 3.5.

L'ordre de préférence suivant $abc \succ ab\neg c \succ a\neg b\neg c \succ a\neg bc \succ \neg a\neg b\neg c \succ \neg a\neg bc \succ \neg abc \succ \neg ab\neg c$ est compatible avec N. Par exemple $a\neg b\neg c$ est ordonnable de manière cohérente par rapport à $a\neg bc$. Aussi $a\neg bc$ est ordonnable de manière cohérente par rapport à $\neg a\neg b\neg c$. En effet ceci assure que nous n'avons ni $N \models a\neg bc \succ a\neg b\neg c$ ni $N \models \neg a\neg b\neg c \succ a\neg bc$. Notons cependant que cela n'est pas suffisant pour dire que l'inverse est vrai, c.-à-d. $N \models a\neg b\neg c \succ a\neg bc$ et $N \models a\neg bc \succ \neg a\neg b\neg c$ Chaque préférence doit être vraie dans tout ordre complet compatible avec N.

Dans cet exemple, il y a seulement deux ordres de préférence compatibles avec N, à savoir l'ordre donné ci-dessus et l'ordre suivant:

$abc \succ ab\neg c \succ a\neg b\neg c \succ \neg a\neg b\neg c \succ a\neg bc \succ \neg a\neg bc \succ \neg abc \succ \neg ab\neg c$

En effet nous avons : $N \models a\neg b\neg c \succ_I \neg bc$ et $N \not\models a\neg bc \succ_I a\neg b\neg c$

- **Complexité des requêtes d'ordres :**

Contrairement aux requêtes de dominance, Les requêtes d'ordre exploitent la structure graphique du graphe et ont l'avantage d'une complexité algorithmique plus raisonnable. La complexité des requêtes d'ordre peut être effectuée en temps linéaire du nombre de variables. Ainsi, notre premier constat est un résultat positif qui indique que les requêtes d'ordre sont effectivement plus identifiables par un temps minimal mais reste le problème de l'adaptation pour les bases de données qui sera étudié dans le chapitre suivant.

Le tableau suivant donne quelques résultats de complexité des requêtes d'ordre

Degré du graphe	**Ordonner**
Direct Tree	O(n)
Polytree ($indegree \leq K$)O(n)	O(n)
Polytree	O(n)
Singly connected ($indegree \leq K$)	O(n)
DAG	O(n)
General case	NP-hard

Figure 3.9 Tableau de la complexité du test de dominance

III.5. Discussion :

L'étude des langages de représentation compacte de préférences est un domaine traditionnel de l'intelligence artificielle, et plus particulièrement des communautés représentation de la connaissance et raisonnement. Et il est souvent difficile de juger précisément de la pertinence d'un langage de représentation compacte, et surtout de son adéquation au problème traité. Néanmoins un certain nombre de critères existe (voir [Coste-Marquis et al., 2004]) pour juger de la pertinence d'un langage :

- Élicitation ;
- Pertinence cognitive ;
- Puissance expressive ;
- Compacité relative ;
- Complexité computationnelle.

Le formalisme des CP-Nets que nous avons présenté tout au long de ce chapitre présent un excellent langage de représentation compacte de préférences.

Les deux premiers critères cités sont très clairement liés à la manière dont l'être humain exprime naturellement des préférences. Le problème de l'élicitation ne se pose pas, car il est plus facile pour un humain d'énoncer une préférence entre une bicyclette rouge et une bicyclette bleue, en revanche, il semble difficile d'affirmer que l'on préfère la bicyclette rouge à hauteur de 8.4 par rapport à la bicyclette bleue.

La puissance expressive est un critère important : les CP-nets sont capables de représenter des préordres possibles mais pas tous. Le gain computationnel obtenu par ce formalisme graphique a une contrepartie en termes d'expressivité.

Les deux derniers critères ont respectivement trait à la complexité du langage de représentation. Si la quantité de mémoire utilisée pour représenter un préordre est

exponentielle, alors le gain espéré de la représentation compacte est perdu. De même, les langages pour lesquels la comparaison de deux alternatives est trop complexe risquent de s'avérer inutilisables en pratique pour la recherche d'une alternative optimale.

Les résultats présentés dans la section.3.4.2 de ce chapitre ont une valeur en eux-mêmes, puisqu'ils permettent une première comparaison des techniques de comparaison d'un point de vue computationnel. Cependant, il faudra proposer des algorithmes adaptés et étudier leurs complexités.

III.6. Cyclique CP-Nets :

D'après la définition des CP-Nets, rien dans la sémantique des CP-Nets le forces à être acyclique. Cependant, le Théorème 3.1 définit l'acyclicité du réseau comme une propriété importante du modèle. Dans [Boutilier et al., 1999], il est montré que si le CP-net N est acyclique, le résultat sera unique. Par contre, en cas d'un graphe cyclique, il peut y avoir plusieurs résultats optimaux, comme il peut n'y en avoir aucun.

On sait qu'un CP-Net est satisfiable si pour chaque alternative o et $o`$, on a $o \succ o`$ ou $o` \succ o$. Le théorème 3.1 déclare qu'un CP-Nets *acyclique* spécifie toujours des ordres préférentiels qui satisfont ce CP-Nets. Malheureusement, La situation est beaucoup plus compliquée pour un CP-Nets cyclique, en plus il n'existe pas des ordres préférentiels qui satisfont le CP-Nets s'il y'a de cycles.

Par exemple, considérez le CP-Net cyclique binaire de la Figure.3.10 (e). Ce réseau est défini sur les variables {A, B}, où A dépend de B et B dépend de A On a :

Dom (A) = {a1, a2}.

Dom (B) = {b1, b2}.

Les TPCs de A et de B sont présentées dans la Figure.3.10 (a), les relations de préférence induites sont consistant (voir le schéma 3.10(b)). Cependant, avec les TPCs présenter dans le schéma 3.10(c), indiquent que les relations de préférence ne sont plus consistantes (voir le schéma 3.10(d)). Cet exemple prouve que la consistance d'un CP-nets cyclique n'est pas garantie, et dépendent de la table TPC. L'analyse systématique de la consistance dans les CP-nets cyclique en général, et dans CP-nets cyclique booléenne en particulier, a été laissée comme problème non résolu dans [Lacroix, 1987].

La figure suivante présente l'exemple du CP-Nets cyclique.

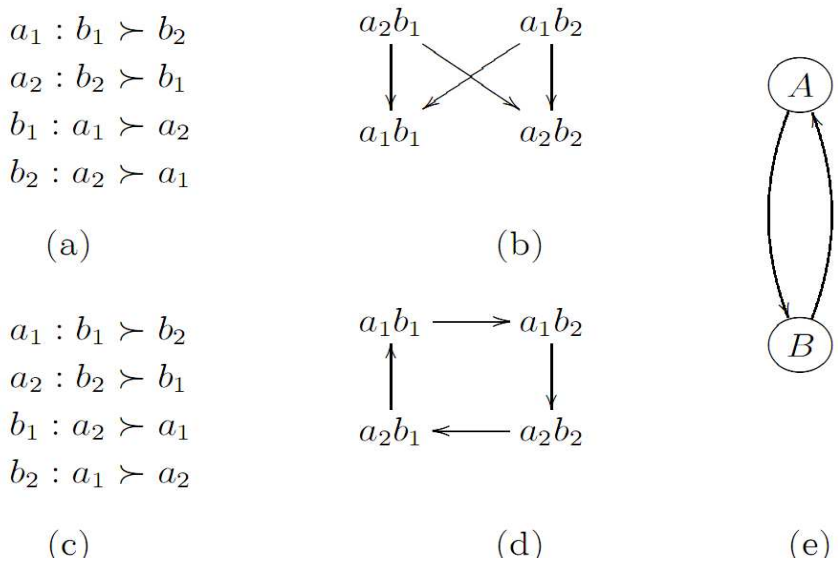


Figure 3.10 *Exemple d'un CP-Nets cyclique*

Le CP-net cyclique contient des cycles, alors il peut exister plusieurs meilleures alternatives. Malheureusement, dés que les critères sont contradictoires, plus rien ne peut être déduit. C'est pourquoi la plupart des travaux sur les CP-nets font l'hypothèse d'acyclicité du graphe. Les travaux de [Boutilier et al., 2004a] montre que dans ce cas précis (acyclicité du CP-net), le réseau est toujours satisfiable. En outre, l'acyclicité du graphe implique des propriétés intéressantes sur la complexité des tâches associées au raisonnement sur la relation de préférence : ainsi, la comparaison d'alternatives et la recherche d'une alternative non dominée sont computationnellement raisonnables [Boutilier et al., 2004b].

Il existe différents travaux récents qui ont été proposés, de manière à augmenter leurs expressivité : Boutilier, Bacchus et Brafman [Boutilier et al ., 2001] proposent le modèle UCP-Net, qui étend le modèle CP-Net en permettant la représentation quantitative d'informations d'utilité plutôt que de simples ordres qualitatifs de préférence. Prestwich, Venable, Rossi et Walsh [Prestwich et al., 2004] ont proposé une nouvelle approche graphique étendant le modèle CP-Net à l'utilisation des contraintes fortes et souples. Brafman et Domshlak dans [Brafman et al., 2002], étendent le modèle CP-Net à la manipulation de la notion d'importance entre variable conduisant au modèle TCP-Net. Finalement, les mCP-Nets proposés dans [Rossi et al., 2004] étendent le formalisme CP-Net pour modéliser et supporter les préférences de multiples agents.

Une autre utilisation intéressante des graphes CP-Nets concerne la présentation adaptative d'informations structurées. Un objectif important est de fournir une personnalisation orientée utilisateur (viewer) de l'information visualisée. Les approches proposées dans [Domshlak et al., 2001], [Brafman et al., 2004] visent respectivement la présentation adaptative des contenus des pages Web, et plus généralement des documents structurés retournés par un provider de contenus. Une présentation initiale est configurée selon les préférences de l'auteur, qui constitue l'expert du contenu. Les approches ainsi proposées se basent sur les graphes CP-Nets pour, d'une part structurer les préférences de l'auteur et offrir ainsi la présentation initiale, et d'autre part, pour assurer l'adaptabilité due la présentation en réponse à la sollicitation de l'utilisateur (viewer) via des algorithmes spécifiques des CP-Nets, pour la recherche de la configuration optimale (optimisation des alternatives).

III.7. Conclusion :

Dans ce chapitre, nous avons présenté le formalisme des CP-Nets, Le formalisme à été proposé afin de traiter les problèmes de décision sans ignorance qui se concentrent sur une seule décision. Il laisse donc de cote les problèmes lies aux décisions séquentielles ainsi que ceux dus aux environnements incertains. Plus précisément, les CP-Nets représente un modèle graphique et qualitatif ce qui permet une formulation naturelle et intuitive et une représentation simple et compacte des préférences.

Nous avons également montré à travers des exemples illustratifs de quelle manière les CP-nets peuvent être utilisés pour la représentation et l'élicitassions des préférences. Ainsi les CP-Nets présentent l'avantage d'être cognitivement pertinent, c.-à-d. une pertinence cognitive très proche de la manière dont les humains expriment leurs préférences.

Nous avons présenté par la suite les différentes techniques d'optimisation et de comparaison des requêtes avec les CP-nets. Comme nous l'avons remarqué, La connaissance dans de nombreux domaines fait intervenir des comparaisons, le choix de la politique de comparaison entre deux alternatives est très important pour le résultat final. Pour permettre de comparer des alternatives entre elles et de déterminer la (ou une des) meilleure(s). Nous avons également aborder le problème de la complexité des réponses

selon deux types de requête, les requêtes de dominance et les requêtes d'ordre. Bien que toutes ces techniques aient pour but de comparer des alternatives sur la base de préférences codées par le CP-net, elles se différencient en plusieurs points : (i) le problème de l'incomparabilité (ii) la complexité computationnelle pour calculer les meilleurs réponses, (iii) l'ensemble des résultats retourner.

Nous avons vu que les résultats présentés par les requêtes d'ordre sont très intéressant non seulement d'un point de vue théorique mais aussi semble particulièrement intéressante pour le problème des réponses ordonnées. Pour cela, nous avons choisi d'adapté cette approche dans le prochain chapitre pour le traitement des préférences exprimées dans les requêtes des utilisateurs.

Chapitre IV

Modélisation & Evaluation des Requêtes à Préférences

IV.1. Introduction

Les requêtes à préférences permettent de prendre en considération des préférences dans les critères de sélection. Nous avons présenté dans le chapitre précédent, le formalisme des CP-Nets (Conditional Preferences Networks), qui présente un langage de représentation compacte et graphique des préférences fondé sur des préférences Ceteris Paribus. On sait qu'un CP-net induit une relation binaire, mais l'interprétation de la relation de préférence sur les alternatives pose alors plusieurs problèmes. L'idée des préférences Ceteris Paribus ou conditionnelles que nous allons introduire dans ce chapitre est de donner un sens précis la notion de préférence pour traitement des requêtes des utilisateurs, c'est à dire, comment interpréter les déclarations de préférence sur les alternatives en terme de relation de préférence $\geq$ sur les alternatives ? On examinera comment l'approche des CP-Nets peut être appliquée dans l'évaluation des requêtes à préférences.

L'objectif de ce chapitre est d'utiliser le critère de comparaison de l'hypothèse Ceteris Paribus pour traiter les requêtes à préférences, ainsi d'avoir par la suite une sémantique destiné à une meilleur interprétation de la notion de préférence dans les requêtes à préférences adressées à une base de données relationnelle. Ce chapitre ne vise qu'à donner un aperçu de notre proposition dans sa globalité. Le développement technique détaille de notre proposition est présenté dans le chapitre suivant.

Ce chapitre est organisé comme suit : En section 4.2, nous décrivons les différentes sémantiques utilisées pour interpréter les requêtes à préférences. La section se décline en deux sous sections. Dans la sous section 4.2.1, nous définissons la sémantique totalitaire. La sous section 4.2.2 est dédiée à la présentation de la sémantique **C**eteris **P**aribus. La section 4.3 présente nos travaux basés sur le formalisme CP-Nets sous la sémantique Ceteris Paribus. Ainsi on Présente notre approche d'évaluation des requêtes CP-Nets. En section 4.4 nous présentons les travaux connexes pour le traitement des requêtes à préférence, afin de mettre en évidence l'intérêt de notre travail.

IV.2. Différentes sémantiques et interprétation des requêtes:

Ces dernières années, le domaine des bases de données a vu émerger des travaux récents portant sur l'expression et l'interprétation des requêtes à préférences dans le contexte du modèle relationnel [Chomicki, 2003; Brafman et al , 2004; Ciaccia, 2007 ; Hadjali et al., 2008], L'objectif général de ces travaux est, d'une part de permettre une meilleur interprétation des requêtes exprimées par l'utilisateur sous forme de préférences contextuelles naturelles, d'autre part dans l'évaluation de cette requête à préférence ,dont l'objectif de calculer la pertinence des tuples pour la requête, pour produire un ensemble de réponses ordonnées.

Différentes propositions ont été faites pour introduire de nouvelle sémantique lors de l'évaluation des requêtes à préférences, On peut distinguer deux approches générales selon qu'on tient compte ou pas du sous-ensemble d'attributs concernés par la requête. Cette préoccupation est illustrée par la requête à préférence suivante : "une **table ronde** sera mieux dans le salon qu'une **table carrée**". Dans le premier cas, ça consiste a ignoré les attributs non pertinents qui ne sont pas présents dans la requête (la taille, la couleur, le bois utilisé, les finitions ou encore le prix). Dans le second cas, l'approche n'exige que toute propriété non pertinente (par rapport a la préférence entre formules que l'on cherche à interprétés) étant égal. Ces interprétations correspondent respectivement, à la sémantique **totalitaire** et la sémantique **Ceteris Paribus**.

Exemple 4.1 : Un agent doit exprimer ses préférences au sujet d'un vol qui doit prendre pour aller à la Mecque : « Je préfère un vol d'air Algérie, plutôt qu'un vol d'Air Egypte ». L'interprétation de cette requête à préférences est la suivante :

« Je préfère un vol d'Air Algérie, plutôt qu'un vol d'Air Egypt. ».

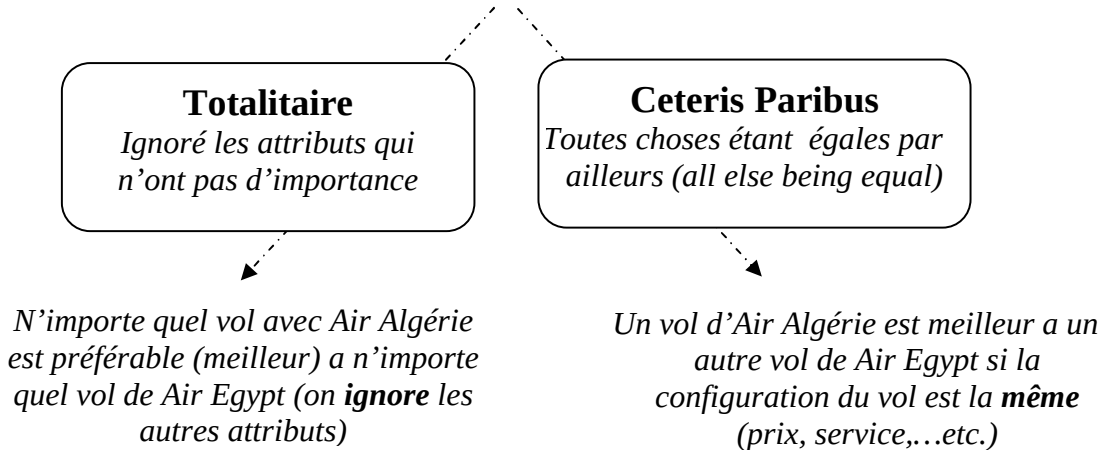


Figure 4.1. Interprétations sémantiques des déclarations de préférences

D'après l'interprétation de la sémantique Ceteris Paribus, on peut remarquer que cette modélisation de préférences est la plus proche de la manière dont les humains expriment leurs préférences. Dans la suite de cette partie, on va montrer que les énoncés préférentiels qualitatifs de type Ceteris Paribus sont d'une part plus fidèle au raisonnement humain et plus conservatrice pour les préférences (c.-à-d. ne conduit pas à des réponses controversées), C'est dans l'objectif de résoudre ce problème, que nous proposons par la suite à travers de notre contribution, un modèle d'évaluation des requêtes à préférences basé sur les CP-Net.

Nous définissons quelques notions en section suivante, avant d'introduire les interprétations et l'évaluation des requêtes de Préférence.

IV.2.1. Notations et définitions préliminaires

Rappelons que le contexte de notre travail est celui des requêtes à préférences adressées à une base de données relationnelle, une approche qualitative adoptée, par exemple dans [Lang, 1996 ; Connan, 1999; Ciaccia, 2007 ; Hadjali et al., 2008], dans laquelle les préférences des utilisateurs sont représentées par une seule relation de préférence (une relation binaire générale sur un schéma).

Formellement, étant donné un schéma de relation R, une requête à préférence Q sur R est composée d'un ensemble $Q = (P_1,\ldots,P_m)$ de déclarations de préférences. Ces déclarations de préférences définissent les relations de préférences $(\varphi_1 :\succ_1,\ldots,\varphi_m :\succ_m)$, respectivement, d'où l'on devrait calculer la relation de préférence globale $\succ_Q$. Par la suite,

Pour toute instance r non vide et finie du schéma R, le système de base de données doit retourner une partie de R contenant, par exemple, les tuples qui sont « optimales », selon la relation globale $\succ_Q$.

Pour illustrer ces concepts, considérons un schéma de relation **Voiture** (catégorie, couleur-ext, couleur- int), et r une instance de voiture décrit dans l'exemple 4.3. Supposons que l'utilisateur exprime la déclaration de préférence *P*.

P : "Je **préfère** les monospaces rouges avec un intérieur blanc **plutôt** que les monospaces blanches avec un intérieur noir ".
La relation de préférence induite par cette déclaration sur R est $\succ_P = \{ t_1 \succ_P t_4 \}$ (alors que tout autres paires de tuples de R sont incomparables selon $\succ_P$).

L'interprétation de tels énoncés ne pose pas de graves difficultés, car elle compare expressément les tuples deux à deux. Cependant, reste la question sémantique « Quel ordre de préférence sur les tuples de la base de données sera induite par une déclaration de préférence *P* »

IV.2.2. la sémantique totalitaire

La sémantique totalitaire avec agrégation semble être le choix de nombreux auteurs dans la communauté des BDs [Rabatti, 1990 ; Connan, 1999]. Les approches qui adoptent cette sémantique tiennent compte que du sous-ensemble d'attributs concernés par la requête et on ignore les autres attributs. Le but de cette section est de montrer que cette sémantique est insuffisante et inadéquate pour les requêtes à préférences conditionnelles. Nous en ferons la preuve avec un certain nombre d'exemples. Tout d'abord, examiner la requête précédente : " Je préfère un vol d'air Algérie, plutôt qu'un vol d'Air Egypte ", les conséquences selon la sémantique totalitaire est que tout choix pour un vol est préféré à tout autre vol à condition que le premier soit avec Air Algérie. Cela devrait déjà disqualifier cette sémantique dans les yeux de beaucoup d'auteurs [Brafman et al., 2004 ; Hadjali et al., 2008], laissez-nous examiner ce qui se passe quand nous avons plus qu'une déclaration de préférence.

Exemple 4.2 :

Considérons les deux déclarations suivantes,

P_1 : " Je préfère les voitures noires aux voitures blanches. " ;

P_2 : " Je préfère les voitures allemandes aux voitures asiatiques.".

Si nous composons les ordres de préférence induite par la sémantique totalitaire de ces deux états en utilisant l'agrégation de l'union, nous obtenons des relations de préférence cycliques et incompatibles: Selon P_1 une voiture asiatique noire est préférée à une voiture allemande blanche :

$$t1 : \text{voit.asiatique} \wedge \text{coul.noire} \succ_{ST} t2 : \text{voit.allemande} \wedge \text{coul.blanche} \Rightarrow t1 \succ t2 .$$

Tandis que selon P_2, on a : une voiture allemande blanche qui est préférée à une voiture asiatique noire :

$$t2 : \text{voit.allemande} \wedge \text{coul.blanche} \succ_{ST} t2 : \text{voit.asiatique} \wedge \text{coul.noire} . \; t2 \succ t1 .$$

$$\Rightarrow t2 \succ t1 \succ t2$$

Comme on est susceptible de faire face à un autre problème lorsque les utilisateurs expriment des préférences sur différents schéma d'attributs. C'est pourquoi les auteurs se sont penchés sur déférents opérateur. Par conséquent, envisager un troisième exemple: P_1 « pour les voiture allemandes, je préfère la '*BMW*' a *VW*," et P_2 - "pour les voitures asiatiques, je préfère Toyota à Isuzu." Ces deux requêtes à préférences décrivent des préférences dans différents contextes. L'intersection des relations de préférence correspondante est vide.

Bien sûr, il existe d'autres opérateurs d'agrégation plus complexes qui sont sémantiquement plus attractif. Par exemple, nous pourrions utiliser l'union ($\bigcup$) sur des contextes disjoints et l'intersection ($\bigcap$) sur des contextes similaires. Malheureusement, cette suggestion n'est pas toujours valable : les contextes peuvent se chevaucher par diverses manières, et la nature des réponses devient de plus en plus compliquée. Une autre solution consiste à utiliser l'union mais avec des priorités. Si nous avons besoin des priorités qui sont définit par l'utilisateur, nous rendons le processus de l'élicitation de préférence encore plus complexe. Si l'utilisateur spécifie des priorités similaires pour les deux préférences sur des attributs différents, on obtient une relation incompatible à nouveau. Une autre approche consiste à essayer d'en déduire des priorités automatiquement, mais ça pose des problèmes au niveau conceptuel, et ça peut avoir de graves coûts de calcul

(complexité non favorable). Le seul aspect rédempteur de la sémantique totalitaire avec agrégation est la facilité de la mise en œuvre de l'approche, l'interprétation totalitaire est plus facile à réaliser algorithmiquement pour comparer les tuples. Mais étant donné ses graves problèmes de sémantique (d'un point de vue cognitif), cela donne une piètre consolation.

Dans la prochaine section, nous présenterons la deuxième sémantique à base de préférences Ceteris Paribus, qui offre une alternative à la sémantique totalitaire, qui est relativement élégante et intéressante d'un point de vue cognitif, en raison de sa nature conservatrice, plus particulièrement, elle évite les pièges sémantiques et les préférences cycliques de la sémantique totalitaire.

IV.2.3. Sémantique Ceteris Paribus :

IV.2.3.1. L'hypothèse Ceteris Paribus

Quand un agent exprime en langage naturel une préférence telle que "*je préfère un appartement au sixième étage à un appartement au rez-de-chaussée* ", il ne veut sans doute pas dire qu'il préfère n'importe quel appartement au sixième étage à n'importe quel appartement au rez-de-chaussée, indépendamment de leurs autres caractéristiques. Ainsi cette préférence n'exclut pas que l'agent préfère toutefois un appartement vaste et luxueux *appartement au rez-de-chaussée* a un studio au sixième étage. Le principe qui est à l'œuvre dans l'interprétation d'une telle préférence est que les alternatives doivent être comparées ***toutes choses étant égales par ailleurs (Ceteris Paribus)*** ou, plus généralement toutes propriétés non pertinentes (par rapport à la préférence entre formules que l'on cherche à interpréter) étant égales.

IV.2.3.2. Logique des préférences du type "Ceteris Paribus"

Les logiques des préférences du type Ceteris Paribus ont été introduites dans [Von Wright, 1963], et revu[1] par [Hansson, 1989, 2001]. Le principe de l'interprétation Ceteris Paribus est fondé sur la constatation suivante : les individus expriment souvent des préférences se référant à des ensembles d'options, et non à des alternatives **isolées**. En termes de formules logiques, cela signifie que les préférences des agents ne sont pas

[1] Hanson a proposé une généralisation fondée sur les fonctions de représentation et qui en a étudié les propriétés logiques.

représentables par des formules complètes, mais par des formules partielles. L'interprétation de la relation de préférence sur ces formules pose alors plusieurs problèmes. Que signifie exactement l'expression « est préférée à », et Comment interpréter la relation de préférence $\geq$ sur les formules en termes de relation de préférence $\succ$ sur les alternatives ?

Répondre à la première question revient à lever des ambiguïtés liées à la traduction d'une préférence « δ est préférée à υ » noté par $\rhd$. Considérons par exemple la préférence précédente : « *je préfère un appartement au sixième étage (formule δ) à un appartement au rez-de-chaussée (formule υ)* », cette déclaration s'écrit formellement par : $\delta \rhd \upsilon$ est La traduction intuitive est bien entendue la suivante : $\delta \wedge \neg\upsilon > \neg\delta \wedge \upsilon$, ou $>$ reste à définir. Un principe particulièrement pertinent est l'interprétation Ceteris Paribus, consistant à interpréter les préférences entre les deux formules δ et υ comme « toutes choses étant égales par ailleurs, je préfère une interprétation satisfaisant $\delta \wedge \neg\upsilon$ à une qui satisfait $\neg\delta \wedge \upsilon$». Ce principe de [Von Wright, 1963] a été réécrit par Hansson [Hansson , 1996 , 2001] en remplaçant $\delta \wedge \neg\upsilon$ par $\delta \backslash \upsilon$: ainsi une relation de préférence $\geq$ satisfait $\delta \backslash \upsilon$ si est seulement si on a $\omega \succ \omega`$ pour toute paire $(\omega, \omega`)$ d'options telles que :

1. $\omega \models \delta \backslash \upsilon$;
2. $\omega` \models \upsilon \backslash \delta$;
3. ω et $\omega`$ coïncident « toutes autres choses étant égales »

En résumé, la traduction d'une préférence $\delta \rhd \upsilon$ sur des formules s'interprète comme $\delta_{\backslash}\upsilon > \upsilon_{\backslash}\delta$, avec $\delta_{\backslash}\upsilon = \delta$ si $\delta \wedge \neg\upsilon$ est inconsistant, et $\delta \wedge \neg\upsilon$ sinon. Ainsi une alternative est préférée à une autre relativement à une préférence > si et seulement si elle vérifie δ et ces deux alternatives sont identiques pour toutes les variables qui ne concernent ni δ ni υ.

Le formalisme des CP-Nets présenté dans le deuxième chapitre illustre le principe de Ceteris Paribus, car il permet une interprétation particulière de la préférence $\rhd$. Il serait intéressant d'étudier la possibilité d'utilisation des graphes CP-Nets, non seulement pour la représentation graphique mais aussi pour l'évaluation de telles requêtes à préférences. En outre, pourrait-on étendre une telle sémantique afin de supporter les requêtes à préférences adressées à une base de données relationnelle ?

La partie qui suit présente l'adaptation des CP-Nets pour le traitement des requêtes dans le contexte des BDs relationnel.

IV.2.3.3. Application au domaine des bases de données relationnelles :

Un des grands avantages du modèle relationnel est sa très grande simplicité. Il n'existe en effet qu'une seule structure, la relation. Une relation peut simplement être représentée sous forme de table. Une (instance de) base de données est un ensemble fini (d'instances) de relations. Le schéma de la base est l'ensemble des schémas des relations de cette base.

Pour mieux illustrer la sémantique Ceteris Paribus, considérer l'exemple suivant : un agent doit exprimer ses préférences au sujet d'une voiture. Ses préférences sont spécifiées sur la catégorie, la couleur extérieure et la couleur intérieure. Ainsi que des dépendances entre les variables.

Exemple 4.3: Soit le schéma de la relation **Voiture** (*Catégorie, Couleur_extér, Couleur_Inter)* d'une base de données Figure 4.2

Les déclarations exprimant les préférences de l'agent sont :

P_1 :	*je préfère les monospaces **rouges** aux monospaces **blanches**.*
P_2 :	*je préfère les voitures de Sport **blanches** aux voitures de Sport **rouges**.*
P_3 :	*dans les voitures **blanches**, je préfère un intérieur **noir**.*
P_4 :	*dans les voitures **rouges**, je préfère un intérieur **lumineux**.*
P_5 :	*je préfère les **monospaces** aux voitures de **Sport**.*

Figure 4.2. Schéma de dépendance préférentiel.

Notre requête à préférence Q sur R est composé d'un ensemble $Q = (P_1,\ldots, P_5)$ de déclarations de préférence. Ces déclarations préférence sont par la suite définit par des relations de préférence ($\varphi_1 :\succ_1,\ldots,\varphi_5 :\succ_5$), respectivement.

	Catégorie	*Couleur_ extérieur*	*Couleur_intérieur*
t_1	Monospace	*Rouge*	Lumineuse
t_2	Monospace	*Rouge*	Sombre
t_3	Monospace	*Blanche*	Lumineuse
t_4	Monospace	*Blanche*	Sombre
t_5	V. Sport	*Rouge*	Lumineuse
t_6	V. Sport	*Rouge*	Sombre
t_7	V. Sport	*Blanche*	Lumineuse
t_8	V. Sport	*Blanche*	Sombre

Figure 4.3. Une instance du schéma Voiture

Commençons par observer une première déclaration de préférence Q : « *Je préfère les* ***monospace rouges*** *aux* ***blanches*** *Voiture de* ***Sport*** ». D'après le principe de *Ceteris Paribus* (*toutes choses non pertinentes étant égales*) on a : $S_{CP}(Q_1) = \{ t_1 \succ_1 t_7, t_2 \succ_1 t_8\}$. Le tuple t_1 (t_2) est meilleur que le tuple t_7 (t_8) alors que le tuple t_1(t_2) et t_8 (t_7), respectivement sont incomparable, Alors que d'après la sémantique totalitaire On a le résultat suivant :

$$S_T(Q_1) = \{t_1 \succ_1 t_7, t_1 \succ_1 t_8, t_2 \succ_1 t_7, t_2 \succ_1 t_8\}$$

Ce premier exemple montre les relations de préférences induites par les deux sémantiques pour une seule déclaration de préférence (on parcourt les n-uplets de la relation Voiture, et on fait une comparaison pour chaque n-uplet). Même si l'interprétation selon la sémantique totalitaire semble plus simple pour une seule déclaration, ça devient plus compliqué pour plusieurs déclarations de préférences. Reconsidérer notre exemple : on a la requête $Q = (P_1,\ldots, P_5)$. La requête Q est composée de plusieurs déclarations de préférence :

Q = {(P_1) : monospaces $\wedge$ rouge $\succ_{ST}$ monospaces $\wedge$ Blanche,
(P_2) : V. sport $\wedge$ Blanche $\succ_{ST}$ V. sport $\wedge$ rouge ,
(P_3) : Blanche $\wedge$ int. Noir $\succ_{ST}$ Blanche $\wedge$ int. lumineux
(P_4) : rouge $\wedge$ int. Lumineux $\succ_{ST}$ rouge $\wedge$ int. Noir
(P_5) : monospaces $\succ_{ST}$ V. sport }

Selon P_1 on a : $t_1 \succ t_3, t_1 \succ t_4, t_2 \succ t_3, t_2 \succ t_4$

Selon P_2 on a : $t_7 \succ t_5, t_1 \succ t_6, t_8 \succ t_5, t_8 \succ t_6$

Selon P_3 on a : $t_4 \succ t_7, t_4 \succ t_3, t_8 \succ t_7, t_8 \succ t_3$

Selon P_4 on a : $t_5 \succ t_6, t_5 \succ t_2, t_1 \succ t_6, t_1 \succ t_2$

Selon P_5 on a : $t_1 \succ t_5, t_2 \succ t_5, t_3 \succ t_5, t_4 \succ t_5, t_1 \succ t_6, t_2 \succ t_6, t_3 \succ t_6, t_4 \succ t_6,$
$t_1 \succ t_7, t_2 \succ t_7, t_3 \succ t_7, t_4 \succ t_7\ t_1 \succ t_8, t_2 \succ t_8, t_3 \succ t_8, t_4 \succ t_8$

d'après la requête à préférence globale Q on aura la relation suivante

$$\succ_Q = \succ_{P_1} \cup \succ_{P_2} \cup \succ_{P_3} \cup \succ_{P_4} \cup \succ_{P_5}$$

Ainsi notre première constatation pour la sémantique totalitaire est confirmée, par ce que les préférences induit sont cycliques et inconsistantes, c.-à-d. :

$t_8 \succ t_3 \quad \approx \quad t_3 \succ t_8$

$t_5 \succ t_2 \quad \approx \quad t_2 \succ t_5$

Tandis que l'union des relations de préférence $\succ_1 \ldots \succ_5$ sous la sémantique totalitaire est problématique, comme on a vu dans cet exemple. L'union des relations de préférence $\succ_1 \ldots \succ_5$ sous la sémantique Ceteris Paribus est plus efficace. Les relations de préférence sont graphiquement représentées dans la figure 4.4 (b). Les nœuds représente les tuples $t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8$, et les liens entre les tuples définissent pour chaque déclaration des préférences P_i, $1 \leq i \leq 5$, Un lien de t vers t` signifie qu'il y'a une relation de préférence entre t et t', tel que $t \succ_i t$`.La figure 4.4 (a) illustre le CP-Net correspondant à l'interprétation de l'énoncé précédent :

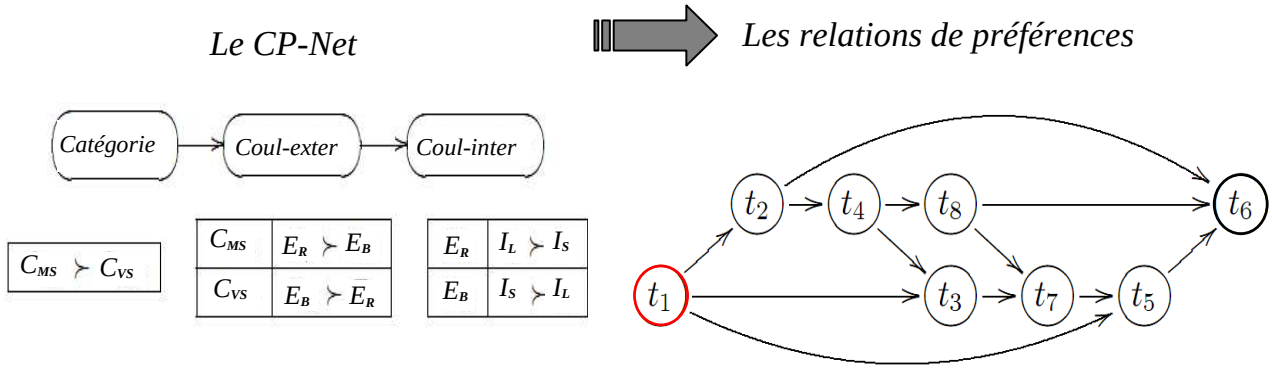


Figure 4.4. - (a) : le CP-Nets des requêtes à préférences $P_{1...5}$, - (b) : Graphe de préférences induit sous la sémantique Ceteris Paribus.

Le résultat de la relation de préférence globale est indiqué par la fermeture transitive du graphe. La propriété de transitivité quant à elle recouvre le fait que le tuple t_1 est préféré au tuple t_2 et t_2 est préféré à t_4, alors t_1 est préféré à t_4.

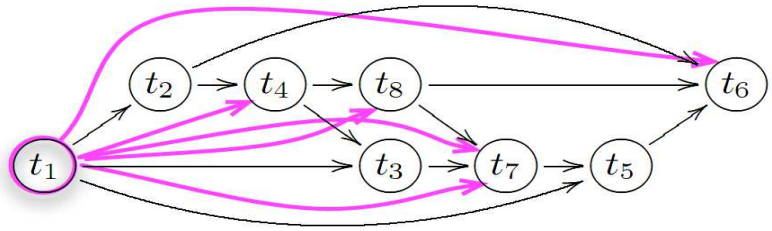


Figure 4.5. → La fermeture transitive du graphe de préférences.

Dans ce chapitre, nous avons présenté une nouvelle sémantique qui mis en valeur les requêtes à préférences, la sémantique Ceteris Paribus représente la façon la plus simple pour décrire les préférences des utilisateurs. À ce stade, nous espérons avoir convaincu que sémantiquement, il n'ya qu'une seul adéquate option - la sémantique Ceteris Paribus.

IV.3. Le modèle propose dans les grandes lignes

IV.3.1. Requête à préférences : un modèle basé sur les CP-Nets

Les préférences conditionnelles constituent la forme la plus usuelle et la plus intuitive des préférences humaines. Ces préférences ne sont pas spécifiquement prises en charge dans les requêtes adressées aux bases de données relationnelles. Il est certes possible de les traduire dans une formulation booléenne dans laquelle la sémantique totalitaire est utilisée pour traduire l'ordre de préférences. Cependant, comme nous l'avons montré en section 4.2, il n'existe qu'une seule sémantique adéquate, la sémantique Ceteris Paribus pour interpréter correctement de telles préférences (conditionnelles), et une approches adoptent la sémantique *totalitaire* peut conduire à des contradictions. C'est dans l'objectif de résoudre ce problème, que nous proposons à travers la présente contribution, une approche basée sur les CP-Nets pour les requêtes à préférences dans le modèle relationnel. En particulier, nous proposons :

- Une approche de représentation CP-Net de requêtes préférentielles exprimant des préférences qualitatives de l'utilisateur,
- Une approche d'optimisation et d'évaluation des requêtes CP-Nets.

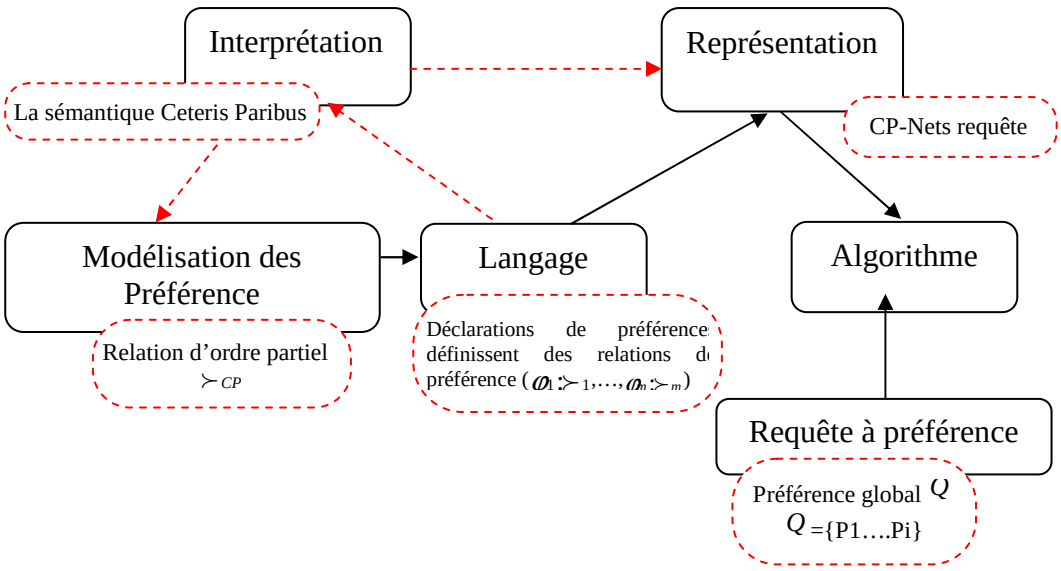


Figure 4.6 Requêtes à préférences : un modèle basée sur les CP-Nets

IV.3.2. Représentation CP-Net des requêtes préférentielles

Une requête à préférence est exprimées en générale dans un langage naturel sous forme de déclarations de préférences, les déclarations de préférences sont alors interprétées par la suite pour spécifier l'ensemble des attributs sur lesquelles vont porter ses préférences. Chaque attribut est défini sur son propre domaine de valeurs. Pour chaque attribut donné *Ai*, l'utilisateur doit spécifier toutes ses dépendances préférentielles, ainsi que l'ordre de préférences correspondant sur Dom(*Ai*).

Cette description est utilisée pour construire le CP-Net requête : Dans le contexte des bases de données les nœuds du CP-Nets correspond aux attributs du schéma, et un arc entre A_1 et A_2 définit la dépendance préférentielle. La table TPC traduit l'ordre des préférences sur le domaine des attributs.

Une représentation des déclarations de préférences de la requête (syntaxique, sémantique) est ainsi construite. Ainsi le CP-Nets requête N_Q correspond aux déclarations de préférences P_i , qui sont traduit en relations de préférence φi sur un schéma relationnel IR, puis ils sont définie en utilisant une formule de préférence $\succ_{CP}$ qui est un ordre partiel strict (sémantique Ceteris Paribus).

Proposition : Un CP-Net sur un schéma de relation R est un couple (G,C), ou G est un graphe dirigé dont l'ensemble de sommets $\{A_1, A_2, A_3, \ldots A_n\}$ est isomorphe à R , et C est un ensemble de tables de préférences conditionnelles qui définissent pour chaque attribut A du CP-Net une table de préférences conditionnelles ($TPC(A_i)$) spécifiant un ordre de préférence total.

La figure 4.4 illustre le CP-Net correspondant à la requête (exemple 4.3). Les variables concernées sont Voiture, Extérieur et Intérieur telles que :

Les valeurs (C_{MS}, C_{VS}), (E_R, E_B) et (I_L, I_S) représentent respectivement les domaines (**M**onospace, **V**oiture_de_**S**port), (**R**ouge, **B**lanche), (**L**umineux, **S**ombres) des attributs de la Catégorie, Couleur_Extérieure, et Couleur_Intérieure. Comme, dans cet exemple particulier on a R = D (R), la Figure 4.5 peut être considérée comme une illustration graphique d'une relation, la fermeture transitive du graphe nous donne exactement la relation de préférence induite par N_Q sur R: Un Arc dans ce graphique réalisé par un tuple ti à un tuple tj indique que la préférence pour les ti sur tj peut être déterminée directement à partir de l'un des TPC dans le CP-net. Par exemple, le fait que $t_4 = C_{ms} \wedge E_R \wedge I_S$ est préféré a $t_3 = C_{ms} \wedge E_R \wedge I_L$ (comme l'indique l'arc qui existe entre eux) est une conséquence directe de la

sémantique Ceteris Paribus, Ces informations sont inscrites dans la table de préférence conditionnel de l'attribut Couleur_Intérieure. La TPC(I) définit un ordre total sur les évaluations de l'attribut Couleur_Intérieure.

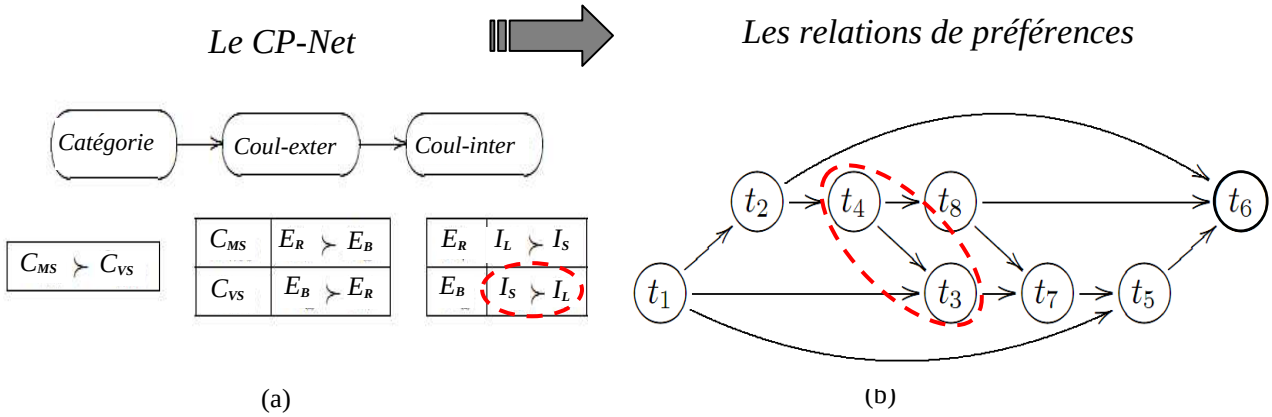


Figure 4.7. L'élicitation des préférences pour la TPC(I)

IV.3.3. Optimisation et évaluation de la requête :

L'objectif de toute modélisation des préférences est de permettre d'optimiser et de comparer des alternatives entre elles et de déterminer la (ou une des) meilleure(s). L'optimisation à pour but de choisir une meilleure solution pour répondre à une requête CP-Nets donnée. C'est une étape complexe et importante qui détermine les performances de notre approche.

Comme nous avons spécifié précédemment, l'idée principale de la sémantique Ceteris Paribus est de faciliter l'écriture et l'interprétation des requêtes de l'utilisateur en les enrichissant avec ses préférences et ses exigences d'une façon simple. Etant donné que la requête à préférence induit un CP-Net acyclique, reste à trouver un plan optimal pour la requête obtenue. Dans ce contexte, on examinera comment l'approche des *CP-Nets* peut être appliquée pour trouver une meilleure solution à ce problème.

Dans un cadre plus général, Ils existent deux approches d'optimisation avec les CP-Nets. Selon le type de stratégie de recherche utilisée pour la comparaison, soit par les requêtes de dominance, ou par les requêtes d'ordre. Rappelons que notre objectif est de pouvoir classer les n-uplets à partir de plusieurs préférences élémentaires. Il s'agit de traduire et de réécrire la requête, de choisir des operateurs appropriés et les algorithmes implémentant ces opérateurs. Afin de mettre en œuvre cette idée et pour satisfaire nos objectifs tout en restant dans le cadre de nos hypothèses, nous pensons qu'il faut (1)

exploiter l'aspect graphique du CP-Nets requête afin de départager le plus d'alternatives possible et rester le plus proche de la requête initiale. Plus précisément, Identifier le CP-Nets correspondant à la requête et spécifier ces TPCs, puis identifier les nœuds Ai (tout les attributs de R) de l'instance de IR concerné par les relations de préférences et l'ensemble des tuples t[Ai] t`[Ai] candidates, (2) ensuite appliquer le principe des requêtes d'ordre pour chercher la meilleure alternative pour le CP-nets requête N , l'idée consiste intuitivement à parcourir le graphe de haut en bas (c'est-à-dire des parents aux descendants), en instanciant chaque variable à sa valeur préférée selon l'instanciation de ses parents. Ainsi on peut aisément déterminer la meilleure alternative sur les ordres de préférence qui satisfont les CP-Net. le présent travail est une continuité du travail proposé dans [Brafman et al., 2004]. Nous partons du constat que la requête à préférences de l'utilisateur forme un CP-nets acyclique, ensuite appliquer un nouveau operateur ORD proposé par les auteurs, qui permet l'ordonnancement et la sélection des éléments préférés d'une relation (R) par rapport à la requête CP-Nets ($\succ_{CP}$), et finalement (3) développer et appliquer un algorithme qui correspond le mieux à notre approche sur cette relation et retourner les tuples préférés par un ordre décroissant de préférences. Notre approche apporte deux principales contributions, à savoir (1) une extension et un raffinement de l'operateur ORD et un algorithme associé.

IV.4. Travaux connexes et intérêts de l'approche

Ces dernières années, le domaine des bases de données a vu émerger des travaux visant à rendre plus flexibles les systèmes, et ce notamment par un enrichissement des langages de requête au moyen de préférences, dont l'objectif est d'accroître la capacité d'expression des langages de requêtes.

IV.4.1. Rapports avec les travaux existants :

Dans cette section, nous présentons l'idée essentielle des principales approches les plus proches de la nôtre. Il existe dans la littérature de nombreux travaux en rapport avec le notre. Ceux-ci visent généralement soit (1) à utiliser le formalisme des CP-nets pour l'interprétation des requêtes à préférences [Ciaccia, 2007] soit (2) à formaliser les requêtes à préférences logiquement par un operateur à base de comparaisons [Chomicki, 2003 ; Kießling, 2002].

IV.4.2. Travaux en rapport avec le formalisme des CP-Nets :

Dans ce premier cas, Nous montrerons d'abord un travail tout nouveau sur les CP-Nets, celui de Paolo [Ciaccia, 2007]. Une approche qui propose un modèle de représentation des préférences conditionnelles avec le formalisme des CP-Nets «Querying Databases with Incomplete CP-nets», c'est un travail qui manipule généralement le formalisme des CP-Nets dans le cas où les préférences sont **incompletes** (C.-à-d., dans le cas ou les préférences d'utilisateur sont indisponibles).

De même, lorsque Paolo à proposé de modéliser les préférences incomplets avec le formalisme des CP-nets en s'inspirant d'une sémantique autre a la sémantique Ceteris Paribus, il a utilisé la sémantique totalitaire afin d'interpréter les déclarations de préférences des utilisateurs.

Principe :

La proposition consiste à spécifier que lorsque les préférences ne sont pas totalement spécifiées on aura un CP-Nets incomplet ainsi on n'aura pas un ordre total sur les préférences et on sera confronté au problème des réponses incomparables (aucune dominance ne peut être établie).

Définition 4.1 [Ciaccia, 2007] **:** Soit une requête Q = P : $a_{i,1} \succ a_{i,2}$ est une préférences dans la table TPC(A). L'interprétation de Q par la sémantique totalitaire est définit par l'ensemble des paires de tuples suivants :

$$\varphi_{st} = \{((p,a_1,y), (p,a_2,y`))|\ y,y` \in \text{dom}\ (X - P - \{A\})\ \}$$

Ainsi, les tuples classés par φ_{st} sont différent sur la valeur de A et, éventuellement, aussi dans les valeurs des attributs de Y.

De plus, lorsque Ciaccia [Ciaccia, 2007] à proposé de modéliser les déclarations de préférences en se basant sur la sémantique totalitaire, il a certes proposé un opérateur d'agrégation sous le principe de Pareto[2] comme operateur de composition de préférences. Cependant l'utilisation du principe de Pareto nous semble contradictoire avec le principe des CP-Nets, Le formalisme des CP-Net implique que les préférences sur les parents ont une priorité supérieure à celles de leurs descendants.

Enfin, la mise en œuvre de l'hypothèse totalitaire pour le formalisme des CP-Nets ne nous semble pas satisfaisante. Nous avons vu dans la section précédente que la

[2] La composition de préférences de la même importance

formalisation de cette hypothèse pour les requêtes à préférences ne permet pas de manipuler les préférences conditionnelles. De plus, à notre connaissance, les solutions proposées demandent de nombreux efforts pour être mises en œuvre puisqu'elles nécessitent la spécification d'une nouvelle relation d'ordre afin de ne pas écarter les propriétés non pertinentes de la comparaison. Ainsi, il manque encore une étude de leur complexité représentationnelle.

IV.4.3. Travaux concernant les modélisations logiques de la préférence :

En ce qui concerne les travaux relatifs à la modélisation logique de la préférence [Chomicki, 2002, 2003 ; Torlone et al., 2002], il nous semble mieux adapté aux requêtes à préférences. Parmi ces approches on retrouve des opérateurs complexes comme l'opérateur *Best.* Celui-ci permet de sélectionner les meilleurs n-uplets, c'est-à-dire, ceux qui ne sont pas dominés au sens de la relation de préférence $\succ_{CP(Q)}$. Cependant, reste le problème de la complexité du calcul des meilleures réponses de *Best* sous le principe *Ceteris Paribus.*

Principe :

On a vu dans le premier chapitre l'approche logique, les langages à base de logique sont très expressifs et intuitifs, l'idée de base de ce modèle est la dominance, la relation de préférence sur le schéma relationnel est définie en utilisant une formule de préférence qui est un ordre partiel strict, les réponses obtenue sont ceux qui ne sont pas dominés par aucun autre tuple, Il est défini dans le contexte des bases de données relationnelles. Deux tuples qui appartiennent au résultat de l'opérateur ont soit les mêmes valeurs, soit sont indifférents entre eux (incomparable).

Définition 4.2 (BEST) [Chomicki, 2002] Étant donné un schéma de relation R, soit Q un schéma de dépendance préférentiel d'une requête induisant un ordre strict de préférence partielle $\succ_{CP(Q)}$ sur D (R). Étant donné **r** une instance de la relation R, alors pour toute instance non vide et finie de ce schéma l'opérateur BEST renvoie un résultat non vide. BEST (R,$\succ_{CP(Q)}$) contient l'ensemble des n-uplets les plus préférés (les meilleurs n-uplets) et ceux qui ne sont pas dominés au sens de la relation de préférence $\succ_{CP(Q)}$.
L'operateur BEST est défini comme suit:

$$\text{BEST}(r, \succ_{CP(Q)}) = \{t \in R \mid \forall\ t` \in R : t` \not\succ_{CP(Q} t\ \}$$

Ou $\succ_{CP(Q)}$ représente la relation de préférence induite du CP-nets requête, définie sur le schéma de dépendance préférentiel (Q = {P_1, … ,P_m}) sous la sémantique Ceteris Paribus et r une instance de schéma de la relation R.

On a vu dans la section 3.4 du troisième chapitre que la complexité de la dominance d'un CP-Nets varie énormément selon la nature et le degré du graphe, les résultats montrent que la complexité computationnelle peut devenir prohibitive pour l'évaluation des requêtes à préférences , même pour les requêtes qui dépendent d'un CP-Nets acyclique sur les attributs binaires, le problème est NP-difficile, et pour les attributs non-binaires, , elle n'est même pas dans NP. En plus, même si l'opérateur Best contient l'ensemble des n-uplets les plus préférés, ils ne sont pas forcement ordonné (ce qui n'est particulièrement pas intéressant dans le cas où l'ensemble des items satisfaisant la requête est de taille importante).

Les requêtes de dominance nous semblent donc généralement mal adaptes à notre problématique. En effet, à notre connaissance, ils sont tous formalises sur la base du test de la dominance, Or, cette dernière nous apparait manquer en terme de complexité de calcul des meilleures réponses, L'inconvénient de l'operateur BEST se trouve dans la complexité de calcule des meilleurs réponses qui nécessitent beaucoup de temps. C.-à-d. le pire de cas de calcul de BEST n'est pas approprié pour des systèmes de base de données.

IV.4.4. Intérêts :

L'approche de modélisation et d'évaluation des requêtes à préférences proposée et que nous exposons dans ce mémoire dans le prochain chapitre avec plus de détails à de nombreux avantages. En particulier elle est adaptée au modèle relationnel, et permet une représentation aisée et intuitive des requêtes à préférences. De plus, bien que mettant en œuvre l'hypothèse *Ceteris Paribus*, un cadre formel pour gérer les contradictions qui peuvent exister entre les différentes déclarations de préférences, elle permet de départager un grand nombre d'alternatives. Enfin, une fois notre modèle définit, il faut bien sur se poser la question complémentaire de la complexité des requêtes (par exemple, combien coute l'identification d'une solution optimale ?) et celle de l'algorithmique associée.

IV.5. Conclusion :

Les requêtes à préférences sont principalement employées pour pouvoir traduire fidèlement ce que l'utilisateur désire exprimer et, d'autre part, de produire un ensemble de réponses ordonnées. Nous avons présenté dans ce chapitre deux approches pour le traitement de telles requêtes, une approche basé sur la sémantique totalitaire, et une autre basé sur la sémantique Ceteris Paribus. Nous avons montré que la sémantique totalitaire est une sémantique inappropriée pour un tel énoncé, ainsi qu'une portée limitée. Cette limitation est due, d'une part, à l'impossibilité d'interpréter précisément la notion des préférences conditionnelles, et d'autre part, à l'impossibilité d'obtenir des résultats cohérents et ordonner (des résultats contradictoires et incohérentes). Par contre, La sémantique Ceteris Paribus est beaucoup plus attrayant. On s'est attaché à examiner la sémantique Ceteris Paribus pour l'évaluation des requêtes à préférences obtenues, en particulier : i) Il en ressort que cette sémantique permet d'apporter un raffinement de l'ensemble des réponses ii) le fait que les résultats obtenus sont cohérents.

Nous avons également présenté notre approche basée sur les CP-Nets. Cette approche est fondée d'une part sur l'expression de requêtes à préférences traduisant les préférences d'un utilisateur, utilisant l'hypothèse Ceteris Paribus. Lors de l'évaluation des requêtes à préférences, on a examiné comment l'approche des *CP-Nets* peut être appliquée pour l'interprétation des préférences dans les requêtes de base de données.

La suite logique de ce travail est un développement technique détaillé de notre proposition. Il serait aussi intéressant d'aborder le problème de la complexité du calcul des meilleures réponses sous le principe *Ceteris Paribus*.

Chapitre V

Vers une approche fondée sur les CP-Nets

V.1 Introduction :

La modification de l'interprétation sémantique du langage d'expression des requêtes entraînent des changements dans la manière d'exécuter de ces requêtes qui doit prendre en compte les exigences de l'utilisateur. D'un coté, pour choisir l'implémentation qui correspondent le mieux à la demande du client et d'un autre coté, il faut utiliser les algorithmes et les stratégies d'exécution de façon à lui fournir un service de qualité. Notre travail s'inscrit dans cette optique, et nous proposons dans ce contexte une approche d'évaluation des requête à préférences basée sur l'utilisation du formalisme des CP-Nets, ainsi notre but est de fournir des réponses rapides et appropriées à une requête à préférence faite par un utilisateur.

L'approche proposée précédemment vise à étendre la sémantique Ceteris Paribus pour les BDs relationnelle, mais elle ne présente pas des mécanismes pour calculer ou d'avoir les meilleures réponses à une requête donnée Q. Plus particulièrement, on propose dans ce chapitre une procédure plus générale pour l'operateur ORD [Brafman et al., 2004] qui examinera comment on peut avoir les meilleures réponses aux requêtes à préférence avec une complexité théorique plus attrayante (en termes de cout et de pertinence des réponses). Nous proposons par la suite un algorithme qui calcule et mets en ouvre l'operateur ORD. Le but principal de notre algorithme est de trouver les tuples pertinents en réponse à une requête utilisateur sans avoir recoure au test de dominance. Ces tuples sont typiquement retournées sous forme d'une liste ordonnée, où l'ordre est basé sur la table de préférence TPCs. Enfin on abordera également le problème de la complexité du calcul de ces meilleures réponses.

Ce chapitre est organisé comme suit. Nous allons dans un premier temps nous intéresser à l'operateur ORD section 5.2, on y mettra l'accent sur le modèle sémantique de l'operateur pour le traitement des requêtes préférentielles, le traitement du CP-Nets requête et la procédure dédie lors du processus de comparaison des tuples candidat. La section 5.3 développe notre modèle proposé dans le chapitre.3. Notre approche s'articule autour de trois sections. La section 5.3.1 présente une nouvelle procédure pour l'operateur ORD, une version améliorée pour le traitement et la recherche des meilleurs résultats. En section 5.3.2, nous présentons notre algorithme d'évaluation pour formalisme CP-Net sur lequel se basent notre modèles, un exemple illustratif ainsi que quelques résultats de complexité sont donnés. La section 5.3.3 présente des résultats de complexité et enfin nous concluant le chapitre dans la section 5.4

V.2 L'opérateur ORD [Brafman et al., 2004] :

V.2.1 Définition et présentation :

L'operateur ORD est une alternative à l'opérateur BEST, qui se distingue immédiatement: Il est basé sur le mécanisme *des requêtes d'ordre* (voir section II.2.2.1) pour sortir et ordonner l'ensemble des résultats d'une relation R selon la requête à préférence $\succ_Q$, ORD peut être éventuellement étendues pour classer les n-uplets correspondant aux préférences et retourne les k meilleurs. ORD est étroitement liée à la norme de SQL ORDER BY dans laquelle les préférences qualitatives sont utilisées comme mesure de comparaison entre tuples. Formellement, ORD est présenter et défini comme suit:

- ORD (R, $\succ_{CP(Q)}$)

Définition 4.1 (ORD) [Brafman et al., 2004] Étant donné un schéma de relation R, soit Q un schéma de dépendance préférentiel d'une requête induisant un ordre partielle strict de préférence $\succ_{CPQ}$ sur D (R). Étant donné r une instance de la relation R, ORD (R, $\succ_{CPQ}$) contient tous les tuples de r, totalement ordonné tel que, pour tout t, t`∈ R, si t apparaît dans ORD ($R$, $\succ_{CPQ}$) avant t` alors on a t` $\not\succ_{CPQ}$ t.

$$\text{ORD}(R, \succ_{CPQ}) = \{ t >> t` \mid t, t` \in R : t` \not\succ_Q t \}$$

Officieusement, ORD nous offre une vue totale[15] sur la relation de base de données qui soit compatible avec $\succ_{CPQ}$: Si t $\succ_{CPQ}$ t`, alors nous savons que ORD montrera t avant t`, mais si t et t`sont incomparable en fonction de $\succ_{CPQ}$, alors ORD ordonne t et t` de façon arbitraire. On peut observez dans un premier temps, qu'il n'ya pas de différence entre l'ensemble des résultats obtenue par ORD et BEST, puisque les deux opérateurs correspondent à une forme de tri, et la COMPARAISON entre les éléments de la base est une partie essentielle de tout les algorithmes de tri. Mais nous montrons par la suite qu'il y a une différence entre BEST et ORD qui s'avère être cruciale (*la complexité computationnelle*). Dans les chapitres précédents, nous avons montré quelques résultats sur les requêtes de dominances pour un CP-Nets totalement spécifié, un ensemble de tuples totalement ordonnés par rapport à un CP-nets est de **NP-difficile**, alors que les résultats pour les requêtes d'ordre est de **O(n)**, ce qui est plus raisonnable pour une représentation compacte.

Commençons par une observation simple mais très importante qui fournit une distinction principale entre ORD et BEST. Pour ordonner une paire de tuples t et t` uniformément à une relation de préférence $\succ_{CPQ}$, nous pouvons être satisfait en sachant seulement que t $\not\succ_Q$ t` ou t` $\not\succ_Q$ t. On peut déduire que cette information est plus facile a connaitre que de connaitre le rapport exact de préférence entre t et t`. Considérer le lemme important suivant :

Lemme 1 [Brafman et al., 2004]: soit N un CP-net acyclique, et t, t` (t $\neq$ t`) deux affectations de préférences complètes sur les variables de N (deux tuples de la relation R). Soit X_i une variable dans N tels que t et t` assignent les mêmes valeurs pour tout les ancêtres de $X_i$ dans N, et des valeurs différentes à $X_i$. Si, on donne un ordre **u** fournie par t (et t`) à **Ui**, si nous avons t[X_i] $\succ_u$ t`[$X_i$], alors nous auront N $\not\models$ t` $\succ$ t. autrement, si t[X_i]et t`[$X_i$] sont incomparable selon U, alors nous avons N$\not\models$ t $\succ$ t`et N $\not\models$ t` $\succ$ t.

Il n'est pas difficile de voir que les conditions présentées par le Lemme1 peuvent être vérifié à temps O(***n***) par un **parcourt** de haut en bas du CP-net. Ce procédé est référencié en tant que procédure de classement de l'opérateur. Le seul point problématique est que le lemme présente une condition qui est suffisante mais non nécessaire pour la

[15] Les tuples de r sont totalement ordonnées, si les préférences de l'utilisateur sont totalement spécifier

vérité de la question $N \not\models t` \succ t$, c.-à-d. le cas ou ni $N \not\models t \succ t`$, $N \not\models t` \succ t$ le problème de l'incomparabilité. Prenons un exemple.

Considérons l'exemple 4.3 du chapitre.3, ou N représente le CP-Net de la requête à préférence sur *R*, la Figure 5.1 représente les préférences induites par ce CP-net et un cas d'incomparabilité.

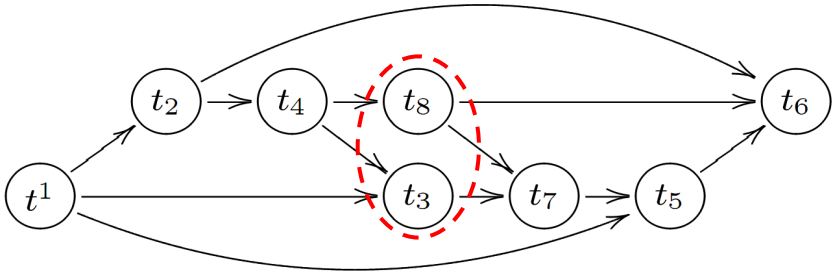


Figure 5.1. *Les préférences induites par le CP-net de l'exemple*
Avec un cas d'incomparabilité

Selon le CP-net de la requête en question, on a les tuples t_3 et t_8 qui représentent respectivement les deux affectations ($C_{MS} \wedge E_B \wedge I_L$), ($C_{VS} \wedge E_B \wedge I_S$) sont incomparables (c'est-à-dire aucun tuple ne peut être révélé préférable à un autre). Cependant, $N \not\models t_3 \succ t_8$ ne peut être déduit en utilisant les conditions du lemme 1, car la ***catégorie*** est la seule variable racine de ce CP-Net, et t_3 assigne une valeur plus préférable que celui attribué par t_8. Par conséquent, ce résultat suivant montre que ce procédé est incomplet et que certaines requêtes de la forme $N \not\models t \succ t`$ peuvent être répondue de manière efficace. Ainsi une amélioration de ce procédé est nécessaire pour avoir un procédé plus complet est une meilleure stratégie de classement. Soit le lemme suivant :

Lemme 2 [Brafman et al., 2004**]** Etant donné N un CP-Nets acyclique, et deux t et t` affectations sur les variables de N, la vérité d'au moins une des requêtes $N \not\models t` \succ t$ ou $N \not\models t \succ t`$ peut être déterminé en utilisant l'operateur ORD..

D'après le lemme précédent, il est possible de fournir une nouvelle procédure une version améliorée de l'opérateur de classement qui définit une extension complète de l'ordre de préférence induite par le CP-net, et son exactitude est affirmée dans notre approche.

De toute évidence, ORD garantit d'afficher n'importe quel élément dans BEST avant tous les éléments qui les domine. Mais, ORD pourrait aussi classer quelques tuple dans BEST dernière si ce tuple (selon la relation de préférences) ne domine pas tout tuple autre. Autre avantage de l'operateur ORD, il permet de retournée les k meilleurs n-uplets correspondant a la requêtes à préférences, pour plus de détails voir l'algorithme proposé dans la section suivante.

V.3 Approche proposée pour notre modèle

Rappelons que notre objectif central est de répondre efficacement à des requêtes à préférences avec plus de pertinences pour les résultats obtenus (une meilleure intégration des préférences d'utilisateur) et une moindre complexité possible (un calcul plus favorable). Nous avons présenté dans le chapitre précédent, notre modèle d'évaluation des requêtes à préférences basé sur la sémantique Ceteris Paribus. La représentation de la requête à préférence est formalisée en graphe CP-Nets. Par ailleurs, notre premier objectif de notre approche proposée et atteint, l'intérêt de ORD se justifie lorsque le CP-Nets (acyclique) délivre un ordre partiel de la requête à préférence, car Celui-ci permet de sélectionner les meilleurs n-uplets.

La partie qui suit, présente notre seconde contribution à travers une étude plus détaillée de notre modèle ainsi que de présenter les mécanismes de classement proposé basé sur le modèle des CP-Nets. L'intérêt majeur de notre modèle est, d'une part, de pouvoir traduire fidèlement ce que l'utilisateur désire exprimer et, d'autre part, de proposer une nouvelle approche pour le traitement du CP-Nets requête, notre approche est un raffinement, bien plus satisfaisant, dans laquelle nous proposons une nouvelle procédure de classement pour l'opérateur ORD adaptée pour produire un ensemble de réponses ordonnées (ce qui est particulièrement intéressant dans le cas où l'ensemble des items satisfaisant la requête est de taille importante).

Dans un second temps, on va présenter un algorithme qui correspond le mieux à notre approche. Enfin, dans un troisième temps, nous terminons ce chapitre en donnant la complexité du calcul des meilleures réponses.

V.3.1 Procédure d'ordonnancement de l'opérateur ORD

Nous avons montré dans la partie précédente de quelle manière ORD arrive à faire un traitement sur le CP-Nets requête construit, ainsi les CP-Nets peuvent être utilisés pour le classement des tuples d'une relation r selon la relation de préférences $\succ_{CPQ}$. Cependant, la procédure dédiée n'est pas complète comme on a vu et peut être moins efficace.

notre procédure proposée pour l'opérateur ORD est plus générale et présente une étude plus complète du processus d'élicitation, ainsi elle arrive à faire un meilleur classement pour une requête à préférences $\succ_{CPQ}$. Notre procédure est fondée principalement sur trois étapes :

1. Analyse et Réécriture de la requête ;
2. Traitement de la requête CP-Nets;
3. Génération d'un plan d'exécution.

V.3.1.1 Analyse de la requête :

Pour toute instance r non vide et finie du schéma R, l'utilisateur doit préalablement spécifier l'ensemble des attributs A_i sur lesquelles vont porter ses préférences par les déclarations de préférences P_i, Pour formuler sa requête. Chaque attribut est défini sur son propre domaine de valeurs. Pour chaque attribut A_i, l'utilisateur doit spécifier toutes ses **dépendances** préférentielles, ainsi que l'ordre de préférences correspondant sur Dom(A_i). Cette description est utilisée pour **construire** le **CP-Net** : les nœuds du CP-Net sont les attributs sur lesquelles portent les préférences utilisateur, les liens entre les nœuds définissent les dépendances préférentielles spécifiées par l'utilisateur. L'ordre de préférences sur un domaine de valeurs est traduit en table TPC.

Reconsidérer l'exemple.4.3 du 3^{éme} chapitre le résultat de cette phase est une représentation et interprétation logique des déclarations de préférences sous le principe *Ceteris Paribus* .

$Q = \{ P_1, P_2, P_3, P_4, P_5 \}$

d'après la requête à préférence globale Q on aura la relation suivante :

$$\succ_Q = \succ_{P1} \cup \succ_{P2} \cup \succ_{P3} \cup \succ_{P4} \cup \succ_{P5}$$

Notre requête à préférence Q sur R est composée d'un ensemble $Q = (P_1, \ldots, P_5)$ de déclarations de préférences. Ces déclarations de préférences sont transformées par des relations de préférences ($\varphi_1 :\succ_1, \ldots, \varphi_5 :\succ_5$), respectivement.

Q = {(P_1) : monospaces ∧ rouge $\succ_{CP}$ monospaces ∧ Blanche,

(P_2) : V. sport ∧ Blanche $\succ_{CP}$ V. sport ∧ rouge ,

(P_3) : Blanche ∧ int. Noir $\succ_{CP}$ Blanche ∧ int. lumineux

(P_4) : rouge ∧ int. Lumineux $\succ_{CP}$ rouge ∧ int. Noir

(P_5) : monospaces $\succ_{CP}$ V. sport }

Ce plan initial est transformé en un plan équivalent et l'interprétation de ce plan est représenté par le formalisme des CP-Nets par un graphe, ou chaque attribut (nœud) A_i dans le graphe est annoté par la table de valeurs TPC.

Ainsi la requête à préférences Q est représentée par le CP-net N qui exprime les préférences sur une tenue de soirée. Le CP-Nets N est présenté comme suit :

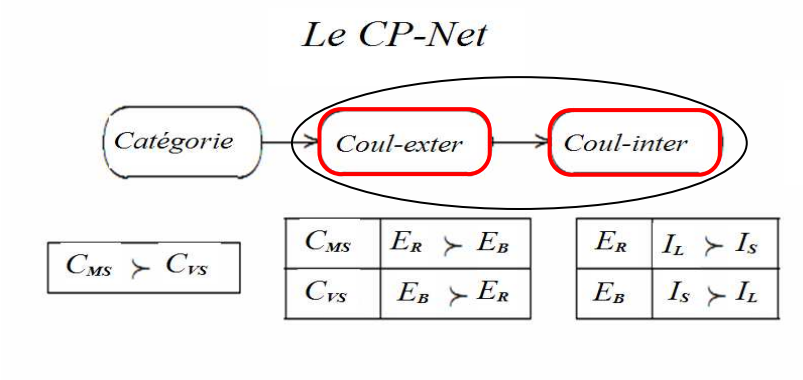


Figure 5.2 - Analyse de la requête CP-Nets

La requête CP-Net est ensuite analysée, premièrement, un balayage séquentiel du graphe est utile dans un premier temps, c.-à-d. un parcours du graphe, nœud par nœud (attribut par attribut). Le parcours est une étape fondamentale pour l'évaluation du CP-Nets requête. On a besoin du parcours de graphe pour connaitre les variables $VR_{t,\,t`}$ qui représentent les attributs préférentiellement dépendant sous l'hypothèse Ceteris Paribus.

Le résultat de cette phase est l'ensemble des variables $RV_{t,\,t`}$ réalisé par le balayage du CP-Nets requête N de haut en bas déjà effectué l'hors de l'étape précédente. Ainsi tous les n-uplets d'une table sont examinés lors de l'opération par un accès séquentiel.

Le résultat de cette phase pour notre exemple nous donne l'ensemble suivant :

$RV_{t,\,t`}$ = { E , I }

Rappelons que notre but est de pouvoir répondre à toute requête sur la **comparaison** de deux tuples quelconque (t, t`) sans avoir recours à la dominance, donc l'objectif visé consiste à obtenir une topologie totale sur les attributs de l'instance r de la relation R et de retourner les n-uplets qui satisfaisant seulement les contraintes de la requête à préférences qui sont délivrés $\succ_Q$.

V.3.1.2 Traitement de la requête C-Nets:

Le traitement de la requête CP-Nets $\succ_Q$ suit les deux étapes suivantes :

1. Obtenir un ordre topologique $\rhd$ du CP-Nets requête :
2. Traitement des TPCs du CP-Nets $\succ_Q$.

1. Obtenir un ordre topologique $\rhd$ du CP-Nets requête :

Une fois les relations de préférences sur le schéma relationnel sont définies en utilisant des formules de préférence qui sont des ordres partiels stricts, alors pour toute instance non vide et finie de ce schéma. Un ordre topologique du CP-Nets est défini. C'est une extension linéaire de l'ordre partiel sur les attributs de r déterminé par les TPCs du CP-Nets, c'est-à-dire un ordre total compatible avec ce dernier. En d'autres termes, Il suffit d'effectuer un parcours en profondeur du graphe CP-Nets, au cours duquel on empile[16] chaque sommet (attribut) une fois ses parents (successeurs) visités. En dépilant, on obtient un ordre topologique total. Un ordre topologique a comme but la production d'un tri topologique.

L'ordre topologique pour notre requête CP-Nets de l'exemple précédent peut donner la succession des sommets C, E et I. En effet, chaque sommet apparait bien avant ses successeurs.

2. Traitement des TPCs du CP-Nets $\succ_Q$:

Le tri topologique réalisé du graphe CP-Nets l'hors de la phase précédente produit une liste des attributs dans un ordre tel qu'un attribut est affiché avant tous ceux qui peuvent être atteints à partir de lui, Pour chaque attribut $A_i \in N$, et pour chaque affectation de u a U parent de A_i, fixer un ordre total $\succ_u$ de préférence compatible l'ordre de partiel $\succ_u$ c.-à-d.

[16] Ce procédé consiste à rechercher une racine, l'enlever, et répéter l'opération autant de fois que nécessaire.

que $\succ_u$ est un ordre de préférence qui est consistant avec la connaissance exprimée dans la TPC(A_i) du CP-Nets.

Concernant notre exemple, les relations d'ordre total déduit sont :

Prenant l'exemple de l'attribut Couleur_Extérieur on a :

1. Monospaces $\wedge$ rouge $\succ_c$ monospaces $\wedge$Blanche,
2. *V.* sport $\wedge$ Blanche $\succ_c$ V. sport $\wedge$rouge ,

Alors fixer un ordre totale $\succ_c$ consistant avec l'ordre partiel $\succ_c$

1. Monospaces $\wedge$ rouge $\succ_c$ monospaces $\wedge$Blanche,
2. *V.* sport $\wedge$ Blanche $\succ_c$ V. sport $\wedge$rouge ,

V.3.1.3 Génération d'un plan d'exécution :

Dans cette section, nous allons présenter la stratégie d'exécution de notre approche de façon à fournir un service de qualité à l'utilisateur pour classer les n-uplets correspondant aux préférences et retourne les meilleurs. Une fois les deux étapes 1 et 2 achevés, reste à ordonner pour chaque attribut $A_i \in VR_{t,t`}$ les tuples de r selon la relation (globale) d'ordre total $>>$. La relation globale $>>$ qui est construite, à partir de toutes les relations partielles étendues précédemment, pour cela il suffit :

Identifiées et vérifiées pour chaque attribut A_i les n-uplets t et t`qui satisfaisant l'ordre total $\succ_u$ de la phase précédente. c.-à-d. pour chaque relation t $[A_i] \succ_u$ t` $[A_i]$ obtenue de la phase précédente, alors retourné t $>>$ t` autrement si on a t`$[A_i] \succ_u$ t $[A_i]$, alors retourné t` $>>$ t (Donc les alternatives les moins préférées sont celles qui ne satisfont aucune TPC).

Sinon ordonner tout les attributs de $VR_{t,t`}$ avec respect à la topologie d'ordre $\triangleright$ réalisé l'hors de la première phase, commençant par le premier attribut de l'ensemble $VR_{t,t`}$ si on a t $[A_i] \succ_u$ t` $[A_i]$ obtenue de la phase précédente, alors retourné t $>>$ t` autrement si on a t`$[A_i] \succ_u$ t $[A_i]$, alors retourné t` $>>$ t

C'est sur cette relation globale $>>$ que se base le classement final des résultats. Les résultats obtenus de cette phase pour notre exemple est le suivant :

1. Monospaces $\wedge$ rouge $\succ_{Cms}$ monospaces $\wedge$Blanche,

$t_1 >> t_3$; $t_1 >> t_4$; $t_2 >> t_3$; $t_2 >> t_4$;

2. *V.* sport $\wedge$ Blanche $\succ_{Vs}$ V. sport $\wedge$rouge ,

$t_7 >> t_5$; $t_7 >> t_6$; $t_8 >> t_5$; $t_8 >> t_6$;

3. rouge $\wedge$ Lumineuse $>_{Er}$ rouge $\wedge$ Sombre,

$t_1 >> t_2$; $t_5 >> t_6$;

4. Blanche $\wedge$ Sombre $>_{Eb}$ Blanche $\wedge$ Lumineuse,

$t_4 >> t_3$; $t_8 >> t_7$;

5. monospaces $>_C$ V. sport }

$t_1 >> t_5$; $t_1 >> t_6$; $t_1 >> t_7$; $t_1 >> t_8$; $t_2 >> t_5$; $t_2 >> t_6$; $t_2 >> t_7$; $t_2 >> t_8$;

$t_3 >> t_5$; $t_3 >> t_6$; $t_3 >> t_7$; $t_3 >> t_8$; $t_4 >> t_5$; $t_4 >> t_6$; $t_4 >> t_7$; $t_4 >> t_8$.

Une fois le classement définit pour chaque attribut de la relation r, l'opérateur ORD fait l'union des résultats partiels et renvoie un résultat final ordonné et non vide. C'est-à-dire les n-uplets les plus préférés au moins préférés de R, ils sont retournés sous forme d'une liste ordonnée, ou tout simplement retournés que les k-meilleures n-uplets de r souhaités. Quand deux tuples ne sont pas comparables par rapport aux préférences, les alternatives les moins préférées sont celles qui ne satisfont aucune TCP, les préférences des fils sont prises en compte pour les comparer. Ceci est un principe de type lexicographique.

Nous présentons dans ce qui suit un résumé de principales étapes utilisées dans notre approche d'ordonnancement d'une instance r d'une relation R par ORD :

Procédure complète :

1. Entrer

Les préférences d'utilisateur $P_1 \ldots P_i$

La relation R= $\{A_1 \ldots A_i\}$

2. prétraitement de la requête

$Q_i = U \varphi_i A_i$

** Les relations de préférences φ_i = *{(Pi) : t* $>_{CP}$ *t` } Ceteris Paribus*

$Q_{CP} = \varphi_1 \cup \varphi_2 \cup \ldots \cup \varphi_i$ ** *utilisé l'union comme operateur d'agrégation*

3. Analyse et Identification du CP-Nets requête

Soit $VR_{t1, t2}$ l'ensemble de touts les attributs A_i de *r*, tel que t_1 et t_2 représente des tuples qui assignent des valeurs différentes pour A_i, mais les mêmes valeurs à tous les ancêtres de A_i en N (en particulier, $u_{t1} = u_{t2}$ pour tous, sauf A_i).

1. Choisir la meilleure alternative pour les nœuds sans parents

2. Pour chaque nœud dont les parents ont été instancies, choisir la meilleure alternative

Identifier $RV_{t, t`}$ par un balayage de haut en bas du CP-Nets requête.

4. Traitement de la requête C-Nets:

Fixer un ordre total, une topologique de classement de *N*. Pour chaque attribut $A_i \in N$, et pour chaque *u* affectations à *Ui*, fixer un ordre total $\succ_u$, compatible avec l'ordre strict partiel $\succ_u$ spécifié par le TPC (A_i).

5. Génération d'un plan d'exécution :

5.1 Pour chaque $A_i \in VR_{t,t`}$, soit ***u**** la tache assignée à Ui faites par t (et t`). Si, pour chaque $A_i \in V_{t,t`}$ on à t $[A_i] \succ_{u*}$ t` $[A_i]$, alors retourné t >> t`. Autrement, si pour chaque $X_i \in V_{t,t`}$ on à t`$[A_i] \succ_{u*}$t $[A_i]$, alors retourné t` >> t.

5.2 Sinon, ordonner $VR_{t,t`}$ avec respect a $\triangleright$. Et choisissez la première variable *Xi* dans le triés $V_{t,t`}$. Si t$[X_i] \succ_{u*}$ t`$[X_i]$, puis retour t >> t`, sinon retourné t >> t` .

6. présenter le résultat final ORD = { t_1 >> t_2 ….>> t_i }

Le résultat est une liste de tuples **pertinents** et **ordonner** pour la requête $\succ_Q$, t_1 : représente la meilleure alternative possible (le tuple le plus préféré) et t_i : représente l'alternative la moins préférable (le tuple le moins préféré).

V.3.2 Algorithme : Calcul de l'opérateur ORD

L'objectif de cette partie est de présenter un algorithme qui correspondent le mieux à notre approche proposé pour l'operateur ORD, Plus particulièrement, nous proposons un algorithme moins complexe et plus optimal pour calculer les meilleures réponses, sachant que la plupart des approches proposées utilisent des algorithmes moins adaptés pour l'évaluation des requêtes à préférences et plus complexe en terme de complexité de calcul des meilleures réponses (le principe de *la dominance* n'est pas utilisé comme stratégie de recherche entre les tuples pour résoudre le problème de la **COMPARAISON**).

Principe :

Notre algorithme à pour but, d'une part, de pouvoir traduire fidèlement les déclarations de préférences et désire exprimer de l'utilisateur et, d'autre part, de produire un ensemble de réponses ordonnées. un prétraitement des relations φ_i est nécessaire sous la sémantique Ceteris Paribus, dont l'objectif est de transformer les déclarations de préférences initial à traiter par des relations de préférences sous forme de formules

équivalente plus simple, et dans un format accepté pour être par la suite traité par notre algorithme de recherche (phase 1).

L'idée consiste à utiliser les tables de préférences conditionnelles TPCs du CP-Nets pour ordonner l'ensemble des tuples de la relation ***r.*** Le traitement de la requête Q_{CP}, revient à traduire les ordres de préférences qualitatives portées par les tables TPCs, en une topologie correspondante est compatible avec le CP-Net requête (phase 2).

une fois un ordre total $>_u$ est associer a chaque TPC(*Ai*) et qui est consistant avec les ordres partiels $\succ$ exprimée dans la TPC(A_i) du CP-Nets, alors :

Soit *t, t`*$\in$ D(R) deux tuples quelconques sur l'ensemble des attributs sur lesquelles les préférences utilisateur sont définies. Chaque tuple *t* correspond à une interprétation de la forme suivante : $t \models \delta \wedge \upsilon$ sachant que δ et υ (l'interprétation Ceteris Paribus) représente des affectations pour les attributs A_i de la relation ***r***. ainsi une relation de préférence $>_u$ satisfait $\succ$ si est seulement si on a $t \succ t'$ telles que :

1. $t \models \delta \wedge \neg\upsilon$;
2. $t' \models \neg\delta \wedge \upsilon$;
3. *t et t`* coïncident « toutes autres choses étant égales ».

Après avoir représenter les déclarations de préférences (P_i) on peut définir les relations de préférence induit comme suit :

$\varphi_{i\ \mathbf{CP}} = \{(P_i) : \delta \wedge \neg\upsilon >_u \neg\delta \wedge \upsilon \}$,

Une fois les déclarations de préférences (P_i) (respectivement la requête à préférences) sont alors traduites en relation de préférences par des expressions booléenne. Chaque requête à préférences étant construite sur l'ensemble des relations de préférences traitées et définies, pour chaque relation de préférences φ_i dans la TPCs on associe une paire de deux sous ensembles t [A_i] et t` [A_i] :

$L(\varphi_i) = \{(t[A_i] , t`[A_i])|\ t \text{ et } t` \in r \}$

Sachant que $t[A_i] = \{t \mid t \in r, t \models \delta \wedge \neg\upsilon\}$ et que $t`[A_i] = \{ t` \mid t` \in r,\ t` \models \neg\delta \wedge \upsilon \}$

t [A_i] et t` [$A_i$] représente la modélisation de notre ordre total de préférences induite de notre ($P_i$) : $t >_u t`$

L'idée principale du prétraitement est de faciliter l'écriture des déclarations de préférences de l'utilisateur pour être facilement traiter par l'algorithme.

- **L'algorithme de calcul de l'opérateur ORD**

Une fois la phase du prétraitement terminée, l'algorithme est très simple. Comme la plupart des algorithmes de tri boucles imbriquées (BNL), il correspond à une boucle de lecture sur les n-uplets de la relation r. L'algorithme à comme entrée un ensemble C_i initialisé par tous les tuples de la relation r (C_i=r), et un ensemble φ_i d'états de préférence comparative de la forme : $(\varphi_i) = \{ (P_i) : \delta \wedge \neg \upsilon >_u \neg\delta \wedge \upsilon \}$, ou $L(\varphi_i)$ représente les deux sous ensembles de chaque relation de préférences φ_i , ensuit ***ORD*** est initialisé par l'ensemble vide (représente le classement des tuples).

La recherche d'une solution optimale n'est pas un problème algorithmique très compliqué tant que la requête à préférences induit un CP-Nets acyclique, ***ORD*** retourne une vue totale sur l'instance r. Dans le cas ou le CP-Nets est complet, le résultat de l'operateur ORD est un ensemble **totalement ordonné** (corps totalement ordonnée) de la relation R, ORD = $\{ t_1 >> t_2 \ldots >> t_i \}$.

Mais lorsqu'il s'agit d'un CP-Nets incomplet, l'algorithme est adapté et on peut toujours avoir les k-meilleurs résultats, ORD = $\{ t_1 >> t_2 \ldots >> t_k \}$, le résultat (non nul) est formé par les éléments qui satisfont le plus grand nombre de relations de préférences.

L'algorithme proposé est fondé sur une optimisation récursive, où, à chaque pas de l'algorithme, on essaie d'avoir le meilleur tuple de l'instance r, le meilleur tuple c'est le premier tuple t_1 . L'exécution d'une requête commence par le parcourt du CP-Nets requête ($V_{Rtt`}$), puis d'identifier les tuple candidates $t$ et $t`$ qui satisfont les relations (φ_i) :

$t[A_i] >_u t`[A_i]$.

A chaque itération un seul tuple est ajouté au résultat ORD = ORD + $\{ t_i \}$

Les grands pas de l'algorithme sont :

1. Afin de connaître les tuples à retourner, ils sont ceux qui n'appartiennent pas à $t`[A_i]$ Outi $\equiv t \mid t \in r, \nexists (t[A_i], t`[A_i]) \in L(\varphi i), t \in t`[A_i]$) (la 2éme ligne de l'algorithme)

2. s'il existe des tuples t qui satisfont cette contrainte, alors pour chaque tuple t dans r on remplace chaque paire ($t[A_i]$,$t`[A_i]$) par $t[A_i] \backslash\{t\}$ *et* $t`[A_i] \backslash\{t`\}$, un seul tuple est ajouté au résultat ORD

3. sinon si Outi = ϕ alors les préférences sont contradictoires.

Ainsi que les relations de préférences qui ont l'ensemble t`[A_i] vide sont éliminées et l'ensemble C_i a son tour est modifié $C_i = C_i \setminus Outi$.

La performance globale de notre algorithme est déterminée par la première étape. Après avoir déterminer les ensembles (t [A_i] ,t`[A_i]) pour chaque relation φ_i , les meilleurs tuples sont extraites de manière simple. Ainsi à la fin de la première itération tous les tuples de l'instance r qui ne satisfont pas la condition doivent être comparés lors de l'itération suivante. L'algorithme s'arrête lorsqu'un l'ensemble C_i ne contient pas de tuple.

En résumé l'algorithme consiste en la recherche du plus **préféré tuple** que l'on va replacer à sa position finale c'est-à-dire en première position, puis on recherche le second plus **préféré tuple** que l'on va replacer également à sa position finale c'est-à-dire en seconde position, etc., jusqu'à ce que la relation *r* soit entièrement trié.

L'algorithme est présenté comme suit :

Algorithme : Calcul de l'operateur ORD

1. Enter : $r, \varphi i , k$

2. Résultat : ***ORD (r)***

3. **Début**

4. $k = 0 , C_i = r$ //

5. **tant que** $C_i \neq \phi$ **faire** :

6. $k = k + 1$ //

7. $Out_k = \{t \mid t \in r , \nexists (t [A_i] ,t`[A_i]) \in L(\varphi i ), t \in t`[A_i] \}$

8. *i.* **Si** $Out_k = \phi$ **alors**

9. Arrêter ***(****relation de préférences inconsistance ou controversées)*

10. //

11. *ii.* **Pour chaque** *tuple t dans* Out_k **faire**

12. *Remplacer chaque* $(t [A_i] ,t`[A_i])$ *dans* $L(\varphi i )$ *par* $t [A_i] \setminus \{t\}, t`[A_i] \setminus \{t`\}$

13. *quand* $t \in t [A_i], t` \in D (P_i)$ *est* $t > t`$

14. *b. supprimer* $(t [A_i] , t`[A_i])$ *avec* $t [A_i]$ *vide*

15. $C_i = C_i \setminus Outi$

16. **Retourner** $ORD_k (r) = (Out_1, \dots , Out_k)$ //

17. **Fin**

Exemple 5.1 : Reprenant la requête précédente, elle est équivalente à :

Q_{CP} = { monospaces ∧ rouge $\succ_{cp}$ monospaces ∧ Blanche,

V. sport ∧ Blanche $\succ_{cp}$ V. sport ∧rouge ,

Blanche ∧int. Noir $\succcurlyeq_{cp}$ Blache ∧int. lumineux

rouge ∧int. lumineux $\succcurlyeq_{cp}$ rouge ∧int. Noir

monospaces $\succcurlyeq_{cp}$ V. sport }

φ_1

$t\overbrace{[A_i]\quad t`[A_i])}$ φ_2 φ_3 φ_4

$L(\varphi_{.i})$ = {({t_1, t_2},{ t_3, t_4}),({t_7, t_8},{t_5, t_6}),({t_4, t_8},{t_3, t_7}),({t_1, t_5},{t_2, t_6}),

({t_1, t_2,t_3, t_4},{t_5, t_6,t_7, t_8})}

φ_5

Notre algorithme commence par la sélection des n-uplets vérifiant la condition 2 puis l'ordonnancement de ces n-uplets suivant l'ordre de sélection et enfin la délivrance des résultats, Le résultat est donné comme suit :

La première itération de l'algorithme nous donne par exemple le tuple t_1 Par ce qu'il y'a que t_1 qui vérifié la condition 2**:**

- 1ére itération Out_1={ t_1 }

$L(\varphi_{.i})$ = {({t_1, t_2},{ t_3, t_4}), ({t_7, t_8},{t_5, t_6}), ({t_4, t_8},{t_3, t_7}), ({t_1, t_5},{t_2, t_6}),

({t_1, t_2,t_3, t_4}, {t_5, t_6,t_7, t_8})}

- 2éme itération Out_2={ t_2 }

$L(\varphi_{.i})$ = {({ t_2},{t_4}), ({t_7, t_8},{t_5, t_6}), ({t_4, t_8},{t_3, t_7}), ({t_5},{t_6}),({t_2,t_3, t_4}, {t_6,t_7, t_8})}

- 3éme itération Out3={ t_4 }

$L(\varphi_{.i})$ = {({t_7, t_8},{t_5, t_6}), ({t_4, t_8},{t_3, t_7}), ({t_5},{t_6}),({t_3, t_4}, {t_7, t_8})}

- 4éme itération Out4={ t_3 }

$L(\varphi_{.i})$ = {({t_7, t_8},{t_5, t_6}), ({t_8},{t_7}), ({t_5},{t_6}),({t_3}, {t_7})}

- 5éme itération Out5={ t_8 }

$L(\varphi_{.i})$ = {({t_7, t_8},{t_5, t_6}), ({t_8},{t_7}), ({t_5},{t_6})}

- 6éme itération Out6={ t_7 }

$L(\varphi_{.i})$ = {({t_7},{t_5}), ({t_5},{t_6})}

- 7éme itération Out7={ t_5 }

$L(\varphi_{.i})$ = {({t_5},{t_6})}

- 8éme itération Out8={ t_6 }

ORD = { t_1 , t_2 , t_4 , t_3 , t_8 , t_7 , t_5 , t_6 }

On rappelle que notre algorithme a une propriété importante, ils garantie que tout les meilleurs tuples non dominés sont ceux trouvés par notre algorithme, l'algorithme les calcule sans recourir au principe de dominance entre toutes les paires de tuples qui n'est pas très attrayants dans la complexité.

V.4 Complexité et optimisation :

La partie précédente concernait exclusivement l'élicitation des préférences et la pertinence cognitive de notre approche proposée pour l'évaluation des requêtes à préférence, ainsi on a présenté la puissance expressive de l'operateur ORD et un algorithme pour le mettre en œuvre. Cependant, il faut bien se poser la question complémentaire de la complexité du calcul des meilleures réponses de l'opérateur sous la sémantique *Ceteris Paribus* (par exemple, combien coute l'utilisation de l'operateur ORD ?) Pour l'identification (des) d'une solution optimale, et pour produire un ensemble de réponses ordonnées (dans le cas où l'ensemble des items satisfaisant la requête est de taille importante).

Ainsi, la complexité du calcul des meilleures réponses est une question cruciale, Par exemple, les résultats présentés dans [Chomicki, 2003] montrent que, même pour une requête à préférences qui induit un CP-Nets acyclique sur les attributs avec des valeurs binaires, le problème est **NP-difficile** (classe la plus ardue), et pour les attributs non-binaires, elle n'est même pas dans **NP**. L'inconvénient se trouve dans la complexité des algorithmes utilisés qui nécessitent beaucoup de temps, et les stratégies de recherche, généralement basés sur la dominance, on peut prendre comme exemple l'operateur BEST, les résultats sont quelque peu pessimiste sous la sémantique Ceteris Paribus : au moins théoriquement, le pire des cas de la complexité de l'operateur BEST est de *O (exp (n). D2)*, où n est le nombre d'attributs et D de la taille de R, Sachant que si la quantité de mémoire utilisée pour représenter un ordre est exponentielle, alors le gain espéré de la représentation compacte est perdu. De même, les langages pour lesquels la comparaison de deux alternatives est trop complexe risquent de s'avérer inutilisables en pratique pour la recherche d'une alternative optimale. Cette complexité qui n'est pas appropriée pour les systèmes de base de données.

Dans la partie qui suit, nous allons nous intéresser à la complexité et à l'optimisation de notre approche proposée précédemment. Le premier problème est lié à *l'optimisation*, c.-à-d. de déterminer s'il existe la meilleure alternative possible. Le second problème concerne *la Complexité computationnelle* : Est-il algorithmiquement difficile de comparer deux tuples quelconque.

V.4.1 Optimisation des résultats:

L'objectif de l'optimisation est la recherche de la meilleure alternative possible. Ainsi notre approche se différencie par la complexité et par la nature du parcours de l'espace de recherche (déterministe. systématique). Du point de vue du fonctionnement, nous nous intéressons à la "forme" du parcours de la stratégie de recherche (tri par sélection).D'après le Lemme1 la complexité du CP-Nets de la requête Q peut être vérifié à temps linéaire O(***n***) par un parcourt de haut en bas du CP-net (du sommet vers les feuilles), donc La complexité de ce problème d'optimisation, pour laquelle on cherche à déterminer s'il existe une solution optimale est un problème **O(n).**

Proposition 5.1 :

Étant donné un schéma de relation R, soit Q un schéma de dépendance préférentiel d'une requête induisant un ordre partiel strict de préférence $\succ_{CPQ}$ sur D (R). Étant donné **r** une instance de la relation R, ORD (R , $\succ_{CPQ}$) contient le résultat optimal de R.

V.4.2 Comparaison :

Rappelons que l'objectif de notre approche ne se limite pas a une seul réponse, mais de produire un ensemble de réponses ordonnées vis-à-vis a l'ordre de préférence induit par le CP-Nets requête. Le tri par sélection est le mécanisme utilisé par notre algorithme de recherche. Il consiste en la recherche du meilleur tuple que l'on va injecter à la première position de ORD, puis on recherche le second tuple (le second plus préféré) que l'on va injecter en seconde c.-à-d. à sa deuxième position, etc., jusqu'à ce que la relation soit entièrement trié. En plus la comparaison entre les tuples utilisés par l'operateur ORD est basé sur la vérification de la relation $t >> t`$, Étend donné que la relation $>>$ est une relation d'ordre total (complète, irrefexive, anti-symmetrique, et transitive) sur D(R), consistent avec la relation de préférence induit par le CP-Nets requête N. alors la complexité de ce problème est de *O (nDlogD).*

Proposition 5.2 : Étant donné un schéma de relation R, soit Q un schéma de dépendance préférentiel d'une requête induisant un ordre partielle strict de préférence $\succ_{CPQ}$ sur D (R). Étant donné **r** une instance de la relation R, alors ORD (R, $\succ_{CPQ}$) peut être calculé en temps O (nDlogD).

En utilisant les résultats précédents, la preuve est simple: Premièrement, l'acyclicité du graphe implique des propriétés intéressantes sur la complexité des comparaisons, la comparaison d'alternatives et la recherche d'une alternative non dominée sont computationnellement raisonnables. Deuxièmement, il est facile de voir que la complexité de notre algorithme est en O(n). Le nombre de comparaisons nécessaires pour un tri est de n(n-1)/2 (où n est le nombre d'attributs). Et finalement, étant donné que $>>$ forme un ordre total. R, alors ORD (R, $\succ_{CPQ}$) peut être calculé en temps O (nDlogD).

V.5 Conclusion :

Nous avons décrit dans ce chapitre, une nouvelle approche adaptée pour les requêtes à préférences basée sur un formalisme unifié, le formalisme CP-Net. l'approche focalise sur deux aspects principaux. Pour se faire, et tout en respectant nos objectifs, Le premier consiste en une nouvelle approche d'évaluation des requêtes CP-Nets basé sur l'operateur ORD plus précisément, Nous avons présenté l'opérateur ORD, un operateur adéquat pour les requêtes a préférences. Nous avons focalisé notre présentation sur la stratégie de recherche de l'operateur ORD, Nous avons également proposé une nouvelle procédure adaptée pour ORD, une approche permet la formulation et le traitement des requêtes à préférences et l'évaluation des requêtes. Le second concerne la mise en œuvre de cette approche, Notre algorithme proposé est utilisé pour gérer à la fois la sémantique Ceteris Paribus et le problème de la complexité du calcul des meilleures réponses, cet algorithme est basé sur le principe de la requête d'ordre qui le rend très attrayant du point de vue de complexité. Plus précisément, le calcul des meilleurs tuples est atteint dans la complexité quasi-linéaire du nombre des préférences de l'utilisateur et le nombre de lignes dans la base de données *O (nDlogD)*.

Conclusion générale

Bilan du travail réalisé

L'intégration des préférences d'utilisateur dans les requêtes adressées aux bases de données permet d'obtenir des réponses plus pertinentes. Ainsi elle immerge comme une approche capitale dans le développement des systèmes du futur (pour que les systèmes d'interrogation deviennent de plus en plus performants.), mais font émerger aussi des nouvelles problématiques d'importance, liées notamment à la formulation et au traitement des préférences exprimées dans les requêtes des utilisateurs. Bien qu'ils existent déjà plusieurs travaux sur ces problématiques exposées ci-dessus, le problème des requêtes à préférences adressées à une base de données relationnelle reste généralement non résolu. Nous avons proposé une étude plus complète sur ce problème, ce travail a été effectué dans un but applicatif afin de répondre à une problématique concrète. Nous pensons avoir en grande partie réussi notre pari initial, qui était l'amélioration de l'évaluation des requêtes à préférences.

Nous avons tenté, dans une première partie de ce mémoire, d'étudier le problème de la formulation et du traitement des préférences exprimées par les requêtes des utilisateurs. Pour ce faire, nous avons commencé par passer en revue plusieurs familles et formalismes permettent d'exprimer les préférences de façon structurée et concise. Comme nous l'avons fait remarquer l'approche explicite (qualitative) est l'approche la plus proche de la manière dont les humains expriment leurs préférences[1]. Nous avons par la suite, présenté deux principales approches de requêtes avec préférences, l'operateur *Winnow* et l'operateur *SKYLINE*.

[1] il est difficile pour l'utilisateur d'exprimer ses préférences au moyen de scores sur les valeurs d'attributs

Par ailleurs, les préférences sur un grand nombre d'alternative peuvent être considérées comme étant conditionnelle. Il est nécessaire de les spécifier de manière plus compacte, en utilisant des représentations logiques ou graphiques. Ainsi, plusieurs approches ont été proposées. Parmi ces approches, nous nous sommes intéressés plus particulièrement à celle basée sur les *CP-Nets* (*Conditional Preferences Networks*). Nous avons également étudié les CP-Nets comme étant un langage d'expression des requêtes à préférences, l'idée principale été de l'amélioration de la représentation des requêtes par la prise en compte des préférences utilisateurs sous l'hypothèse *Ceteris Paribus* (toutes choses étant égales par ailleurs).

Dans la deuxième partie de notre mémoire, nous avons commencé par présenter deux approches d'interprétation et d'évaluation des requêtes à préférence, Peuvent être utilisées pour interpréter les différentes déclarations de préférence qui constitue la requête. Vu que la plupart des approches adoptent la sémantique *totalitaire,* cette sémantique conduit à des évaluations incorrectes. Comme nous l'avons montré, et peut conduire à des contradictions. Nous avons alors opté pour une deuxième approche, une approche fondée sur la logique des préférences du type "***Ceteris Paribus***". L'évaluation des requêtes à préférence à base de préférences Ceteris Paribus, offrent une alternative à la sémantique totalitaire. Nous avons en effet proposé une approche qui définit le formalisme CP-Net comme langage d'expression des requêtes utilisateur portant sur des préférences qualitatives. L'utilisation des CP-Nets offre plusieurs avantages :

1. Les CP-Nets supportent tout naturellement les préférences conditionnelles qualitatives,
2. ils offrent un formalisme graphique simple et intuitif qui permet de structurer ces dernières de manière compacte,
3. le formalisme est relativement élégant et intéressant d'un point de vue cognitif.

La proposition dans sa globalité définit ainsi une nouvelle approche d'évaluation des requêtes à préférences basé sur les CP-Nets. L'approche consiste à :

1. Traduire fidèlement ce que l'utilisateur désire exprimer, ces préférences et exigences ont été traduites dans un langage afin d'être ajoutées aux requêtes et

exécutées. L'approche utilisée ici consiste à faciliter l'écriture des requêtes de l'utilisateur, et à les représenter par un CP-Net requête.

2. Produire un ensemble de réponses ordonnées selon le CP-Net requête déduit, Pour ce faire, nous avons étendu l'operateur ORD, Celui-ci permet d'ordonner l'ensemble des résultats d'une relation R selon la requête CP-Nets $\succ_{Q_{CP}}$. Ainsi on a proposé une nouvelle procédure adaptée pour l'évaluation de la requêtes CP-Nets $\succ_{Q_{CP}}$ Le résultat est une liste de tuples **pertinents** et **ordonnées** pour la requête. La procédure proposée est fondée principalement sur trois étapes :
   a. Analyse et Réécriture de la requête ;
   b. traitement de la requête CP-Nets;
   c. Génération d'un plan d'exécution.

3. la mise en œuvre de cette approche, un algorithme a été proposé pour gérer à la fois les préférences conditionnelles et le problème de la complexité du calcul des meilleures réponses sous le principe *Ceteris Paribus*. notre algorithme a une propriété importante, il garantie que tout les meilleurs tuples non dominés sont ceux trouvés par notre algorithme, l'algorithme les calcule sans recourir au principe de dominance entre toutes les paires de tuples qui n'est pas très attrayants dans la complexité.

Perspectives

Les perspectives pour notre travail se déclinent en deux principaux points, le premier concerne la validation expérimentale de l'approche proposée, la seconde porte sur les améliorations possibles de notre approche.

1. **Validation expérimentale :**

La validation expérimentale de notre approche basée sur les CP-Nets à pour objectif surtout pour montrer la faisabilité de notre approche et d'évaluer la viabilité du modèle et de le comparer par rapport à d'autres approches et modèles de référence. La validation expérimentale de notre modèle proposé, nécessite la construction d'un cadre d'évaluation supportant des requêtes CP-Nets. La construction d'un tel environnement relève d'un travail de recherche à part entière.

2. Améliorations futures

Dans un travail futur, Il serait aussi intéressant de voir dans quelle mesure l'approche des CP-Nets peut être utilisée dans le cas des requêtes à préférences formulées en utilisant des prédicats flous. Les ordres de préférences qualitatifs devraient être traduits en ensembles flous de valeurs d'utilités. Ceci permettrait de fuzzyfier le langage de requête et fournirait le moyen pour une plus large expressivité des requêtes utilisateur. Il est alors possible de prendre en compte des énoncés préférentiels modulés par des opérateurs linguistiques « extrêmes » (à l'exemple de : « je préfère de loin le jus d'orange au jus de pomme »).

Références bibliographiques

[Agrawal et al., 2000] R. Agrawal and E. Wimmers. *A Framework for Expressing and Combining Preferences*. In Proc. SIGMOD, Texas, USA, 2000, pp.297-306.

[Borzsonyi et al., 2001] S. Borzsonyi, D. Kossmann, and K. Stocker, *The skyline operator*, In *ICDE 2001,* pages 421–432, 2001.

[Bouaziz et al., 1997] Bouaziz T., Wolski A., Applying fuzzy events to approximate reasoning in active databases, In Proc.Sixth IEEE Inter. Conf. on fuzzy systems Barcelona, Catalonia, Spain, July1997.

[Boutilier , 1994] Craig BOUTILIER : *Toward a logic for qualitative decision theory*. Dans Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR'94), pages 75–86, 1994.

[Boutilier et al., 1999] Craig BOUTILIER, Ronen I. BRAFMAN, Holger H. HOOS et David POOLE : Reasoning with Conditional Ceteris Paribus Preference Statements. Dans Uncertainty in Artificial Intelligence (UAI'99), 1999.

[Boutilier et al., 2001] Boutilier C.,, Bacchus F., and Brafman R., UCP-Networks : A directed graphical representation of conditional utilities, UAI'2001, p. 56–64.

[Boutilier et al., 2004a] Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. (JAIR) 21, 135-191, 2004.

[Boutilier et al., 2004b] Craig BOUTILIER, Ronen I. BRAFMAN, Carmel DOMSHLAK, Holger H. HOOS et David POOLE : Preference-Based Constrained Optimization with CP-nets. Computational Intelligence, 20(2):137–157, 2004b. Special Issue on Preferences.)

[Brafman et al., 2002] R.I. Brafman and C. Domshlak. Introducing variables importance tradeoffs into CP-Nets. In Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI'02), pages 69 76, 2002.

[Brafman et al., 2004] Brafman, R. I., Domshlak, C.: Database preference queries revisited. Technical Report TR2004-1934, Cornell University, Computing and Information Science, 2004.

[Chengkai et al., 2005] Chengkai Li, K. Chen-Chuan Chang, Ihab F. Ilyas, Sumin Song: RankSQL : *Query Algebra and Optimization for Relational Top-K Queries*. In SIGMOD 2005, pages 131-142, 2005.

[Chevaleyre et al., 2007] Yann Chevaleyre, Ulle Endriss, Sylvia Estivie et Nicolas Maudet : Reaching envy-free states in distributed negotiation settings. Dans Manuela M. Veloso, éditeur : Proceedings of the 20th International Joint Conference on Arti cial Intelligence (IJCAI-07), pages 1239 1244, Hyderabad,India, janvier 2007a. AAAI Press.

[Chomicki, 2002] Querying with Intrinsic Preferences, Jan Chomicki, EDBT, 2002

[Chomicki, 2003] Chomicki, J.: *Preference formulas in relational queries*. ACM Trans. on Database Syst., 28, 1-40, 2003.

[Ciaccia, 2007]Ciaccia, P.: Querying databases with incomplete cp-nets. In: M-PREF 2007. Multidisciplinary Workshop on Advances in Preference Handling (2007)

[Connan , 1999]Connan F., Interrogation flexible de bases de données multimédia, Thèse de doctorat, Université de Rennes I, 1999.

[Coste-Marquis et al., 2004]) Sylvie Coste-Marquis, Jérôme Lang, Paolo Liberatore et Pierre Marquis : Expressive power and succinctness of propositional languages for preference representation. Dans Didier Dubois, Christopher A. Welty et Mary-Anne Williams, éditeurs : Proceedings of the 9th International Conferenceon Principles of Knowledge Representation and Reasoning (KR-04), pages 203 212, Whistler,Canada, juin 2004. AAAI Press.

[Domshlak, 2002] Domshlak, C.: Modeling And Reasoning About Preferences With CP-nets. Thèse de doctorat, Ben-Gurion University of the Negev, Negev, Israel, 2002.

[Domshlak, et al., 2001]C. Domshlak, R. Brafman, and S. E. Shimony. Preference-based configuration of web page content. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pages 1451–1456,Seattle, August 2001.

[Govindarajan et al. 2001] Govindarajan, K., Jayaraman,B., AND Mantha, S. 2001. Preference queries in deductive databases. New Gen. Comput., 57–86.

[Hadjali, 2007] Hadjali, A.: *Requêtes à Préférences & Interrogation Flexible*. Cours de base de données 2007 IRISA/ENSSAT, Université de Rennes 1

[Hadjali et al., 2008]Hadjali, A., Kaci, S., Prade, H.: Database preferences queries - A possibilistic logic approach with symbolic priorities, FoIKS 2008.

[Hansson, 1989] Sven Ove HANSSON : A New Semantical Approach to the Logic of Preference. Erkenntnis, 31:1–42,1989.

[Hansson, 1991] Sven Ove HANSSON : An overview of decision theory. Rapport technique 41, SKN, 1991.

[Hansson, 1996] Hansson, S. (1996). What is ceteris paribus preference. Journal of Philosophical Logic, 25(3) :307–332.(cite aux pages 41, 42)

[Hansson, 2001] Sven Ove HANSSON : The Structure of Values and Norms. Cambridge University Press, 2001.

[Hristidis et al. 2001] HRISTIDIS, V., KOUDAS,N., AND PAPAKONSTANTINOU, Y. PREFER: A system for the efficientexecution of multiparametric ranked queries. In Proceedings of the ACM SIGMOD International Conference on Management of Data. ACM, New York, 259–270, 2001.

[Kießling et al.,, 1994] W. Kießling, G¨ UNTZER, U. 1994. Database reasoning—A deductive framework for solvinglarge and complex problems by means of subsumption. In Proceedings of the 3rd Workshop on Information Systems and Artificial Intelligence. Lecture Notes in Computer Science, vol. 777, Springer-Verlag, New York, 118–138.

[Kießling , 2002] W. Kießling, Foundations of preferences in database systems. In Proceedings of the International Conference on Very Large Data Bases. 2002

[Kießling et al., 2002a] W. Kießling, G. Köstler: *Preference SQL - Design, Implementation, Experiences*. Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[Kießling et al., 2002b] Kießling,W. AND Hafenrichter, B. 2002. Optimizing preference queries for personalized web services. In Proceedings of the IASTED International Conference on Communications, Internet and Information Technology. Also Tech. Rep. 2002-12, July 2002, Institute of Computer Science, University of Augsburg, Germany.

[Kiessling, 2002] W. Kiessling. *Foundations of Preferences in Database Systems*. Proc. Very Large Data Bases, 2002.

[Kostler et al., 1995] G. Köstler, W. Kießling, H. Thöne, U.Güntzer: Fixpoint Iteration with Subsumption in Deductive Databases. In Journal of Intelligent Information Systems, Vol. 4, pp. 123-148, Boston, USA, 1995.

[Lacroix et al., 1987] M. Lacroix, P. Lavency "preferences : Putting More Knowledge into Queries" 13th VLDR Conference, 217-225, 1987.

[Lang, 1996] Jérôme LANG : Conditional desires and utilities – An alternative logical framework for qualitative decision theory. Dans European Conference on Artificial Intelligence (ECAI’96), pages 318–322,1996.(7)

[Lang, 2007] Jérôme LANG : Représentation des préférences, Cours sur les bases de données avancées 2007.

[Larsen, 1999] Larsen H. L., An Approch to Flexible Information Acces Systems Soft Computing, Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.

[Paolo, 2007] Paolo Ciaccia : Querying Databases with Incomplete CP-nets .dand VLDB '07, September 23-28, 2007, Vienna, Austria.

[Prestwich et al., 04] Steve PRESTWICH, Francesca ROSSI, Kristen Brent VENABLE et Toby WALSH : Constrained CP-nets. Dans Italian Conference on Computational Logic, 2004.

[Rabatti, 1990] F. Rabatti, "Retrieval of Multimedia Documents by Imprecise Query Specication" Lecture Notes in Computer Science, 416, pp. 202-218, 1990.

[Rossi et al., 2004] Rossi, F., Venable, K. B., and Walsh, T. (2004). mCP nets : representing and reasoning with preferences of multiple agents. In proceedings of AAAI'04, San Jose, CA, USA.

[Torlone et al., 2002]Torlone, R., Ciaccia, P.: Finding the best when it's a matter of preference. In: SEBD 2002. Proc. 10th Italian National Conference on Advanced Data Base Systems, pp. 347–360 (2002).

[Von Wright, 1963] Georg Henrik von WRIGHT : The logic of preference. Edinburgh University Press, 1963.