N° d'ordre : ANNÉE 2013

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

pour le grade de

**DOCTEUR DE L'UNIVERSITÉ DE RENNES 1**

*Mention : Informatique*

**École doctorale MATISSE**

présentée par


# Katia ABBACI

préparée à l'unité de recherche n° 6074 IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
École Nationale Supérieure des Sciences Appliquées et de Technologie


---

# Contribution à l'interrogation flexible et personnalisée d'objets complexes modélisés par des graphes

**Thèse soutenue à l'Enssat le 12/12/2013**

devant le jury composé de :

**Mohand-Said HACID**
Professeur, LIRIS Univ. de Lyon1 / rapporteur

**Maria RIFQI**
MCF-HDR, LIP6 Univ. de Paris 2 / rapporteur

**Sébastien FERRÉ**
MCF, IRISA Univ. de Rennes1 / examinateur

**Daniela GRIGORI**
Professeur, LAMSADE Univ. Paris Dauphine / examinatrice

**Allel HADJALI**
Professeur, ENSMA de Poitiers / directeur de thèse

**Daniel ROCACHER**
Professeur, ENSSAT de Lannion / directeur de thèse

**Ludovic LIÉTARD**
MCF-HDR, IUT de Lannion / directeur de thèse

**Résumé**

Plusieurs domaines d'application traitent des objets et des données complexes dont la structure et la sémantique de leurs composants sont des informations importantes pour leur manipulation et leur exploitation. La structure de graphe a été bien souvent adoptée, comme modèles de représentation, dans ces domaines. Elle permet de véhiculer un maximum d'informations, liées à la structure, la sémantique et au comportement de ces objets, nécessaires pour assurer une meilleure représentation et une manipulation efficace. Ainsi, lors d'une comparaison entre deux objets complexes, l'opération d'appariement est appliquée entre les graphes les modélisant.

Nous nous sommes intéressés dans cette thèse à l'appariement approximatif qui permet de sélectionner les graphes les plus similaires au graphe d'une requête. L'objectif de notre travail est de contribuer à l'interrogation flexible et personnalisée d'objets complexes modélisés sous forme de graphes pour identifier les graphes les plus pertinents aux besoins de l'utilisateur, exprimés d'une manière partielle ou imprécise.

Dans un premier temps, nous avons proposé un cadre de sélection de services Web modélisés sous forme de graphes qui permet (i) d'améliorer le processus d'appariement en intégrant les préférences des utilisateurs et l'aspect structurel des graphes comparés, et (ii) de retourner les services les plus pertinents. Une deuxième méthode d'évaluation de requêtes de recherche de graphes par similarité a également été présentée pour calculer le skyline de graphes d'une requête utilisateur en tenant compte de plusieurs mesures de distance de graphes. Enfin, des approches de raffinement ont été définies pour réduire la taille, souvent importante, du skyline. Elles ont pour but d'identifier et d'ordonner les points skyline qui répondent le mieux à la requête de l'utilisateur.

**Mots clés :** Interrogation de bases de données, stratégies de recherches dans les bases de données, recherche d'information, théorie des ensembles flous, quantificateurs linguistiques, requêtes, services Web, théorie des graphes.

**Abstract**

Several application domains deal with complex objects whose structure and semantics of their components are crucial for their handling. For this, graph structure has been adopted, as a model of representation, in these areas to capture a maximum of information, related to the structure, semantics and behavior of such objects, necessary for effective representation and processing. Thus, when comparing two complex objects, a matching technique is applied between their graph structures.

In this thesis, we are interested in approximate matching techniques which constitute suitable tools to automatically find and select the most similar graphs to user graph query. The aim of our work is to develop methods to personalized and flexible querying of repositories of complex objects modeled thanks to graphs and then to return the graphs results that fit best the users 'needs, often expressed partially and in an imprecise way.

In a first time, we propose a flexible approach for Web service retrieval that relies both on preference satisfiability and structural similarity between process model graphs. This approach allows (i) to improve the matching process by integrating user preferences and the graph structural aspect, and (ii) to return the most relevant services. A second method for evaluating graph similarity queries is also presented. It retrieves graph similarity skyline of a user query by considering a vector of several graph distance measures instead of a single measure. Thus, graphs which are maximally similar to graph query are returned in an ordered way. Finally, refinement methods have been developed to reduce the size of the skyline when it is of a significant size. They aim to identify and order skyline points that match best the user query.

**Key words:** Querying databases, research strategies in databases, information retrieval, fuzzy set theory, linguistic quantifiers, queries, Web services, graph theory.

# Remerciements

Je tiens à remercier mes directeurs de thèse : Allel Hadjali, Ludovic Liétard et Daniel Rocacher pour leur encadrement et leur conseil précieux tout au long de mon doctorat.

Je souhaite exprimer ma gratitude aux membres du jury pour avoir bien voulu consacrer une partie de leur temps à ma thèse. Je remercie Dr Maria Rifqi et Pr Mohand-Said Hacid d'avoir accepté de rapporter ma thèse, Pr Daniel Grigori et Dr Sébastien Ferré d'avoir accepté de faire partie du jury de ma thèse.

Je remercie également tout le personnel de l'ENSSAT pour leur aide et assistance dans toutes mes démarches administratives.

Je voudrais accorder une place d'honneur dans mes remerciements à ma famille, plus particulièrement, mes parents et mes frères. Leur soutien et encouragement n'a jamais failli au cours de ce long cursus universitaire.

Enfin, je remercie tous mes amis pour m'avoir soutenu durant cette période de thèse.

# Table des matières

# Table des figures

# Liste des tableaux

# Introduction générale

# Introduction générale

## Contexte et problématique

De nos jours, les applications issues de différents domaines requièrent la manipulation et le traitement des informations et des données de plus en plus complexes. Nous pouvons citer, comme exemples, les applications liées à la chimie, à la biologie, au traitement d'images, aux services Web, à la médecine, etc. Des modèles de représentation sont donc nécessaires pour manipuler, stocker et exploiter ces données. Par ailleurs, la qualité des modèles de représentation impacte fortement la pertinence des systèmes exploitant ces données et objets complexes. En effet, les propriétés et les caractéristiques de ces derniers doivent être conservées par ces modèles pour en assurer une meilleure exploitation. À titre d'exemple, le choix d'une modélisation d'un service Web dans un système donné doit être en capacité de sauvegarder ses caractéristiques en termes d'activités le composant, les relations entre ses activités et aussi le flux de contrôle entre ces dernières.

Un des moyens permettant de représenter fidèlement une donnée ou un objet complexe, est la structure de graphe. En effet, les graphes ont la capacité d'exprimer la structure (topologique) de chaque objet complexe, telle que la structure d'une composante chimique, d'une image ou encore d'un document XML. De plus, ils permettent de représenter les relations possibles entres les composants d'un objet, contrairement aux vecteurs dont les informations se limitent aux valeurs des composants de l'objet. Ainsi, la structure de données d'un graphe permet de véhiculer non seulement les informations sur les composants d'un objet et leurs caractéristiques, mais aussi sur leurs agencements topologiques.

Plusieurs travaux ont été menés pour exploiter ce type de structure représentant des objets complexes issus de différents domaines [23, 33, 82, 132]. Une famille de ces travaux a essentiellement porté sur l'opération d'appariement de graphes. Cette opération joue un rôle central, par exemple, dans le contexte de recherche de graphes (parmi une collection ou une base de graphes cibles) qui s'apparient totalement ou partiellement à un graphe requête (représentant l'objet complexe recherché par l'utilisateur).

L'appariement exact vise à rechercher les graphes de la base structurellement identiques au graphe de la requête. Cependant, en pratique, deux graphes sont rarement identiques, même s'ils ont été extraits du même objet. En effet, des différences structurelles peuvent être produites par l'incertitude lors du processus d'extraction de caractéristiques, par des environnements bruités, etc. Par exemple, les graphes représentant des images médicales peuvent être imparfaits dû au dysfonctionnement

des scanners, mauvais réglage d'appareil, ... Pour cela, un appariement approximatif entre graphes a été proposé permettant d'identifier la correspondance entre deux graphes malgré l'existence de différences structurelles mineures. Notons que dans certaines applications, l'appariement de graphes doit tenir compte, en plus de l'aspect structurel, des préférences et des contraintes de l'utilisateur exprimées dans sa requête.

Le grand volume de données stockées dans les bases de données de graphes et la complexité et l'imprécision des besoins de l'utilisateur exprimés via le graphe requête, conduisent souvent à des résultats de nature pléthorique ou vide. Dans le cas des réponses pléthoriques, l'établissement d'un ordre entre ces réponses est crucial pour permettre à l'utilisateur de choisir les meilleures réponses. Quant aux réponses vides, la solution est de retourner à l'utilisateur des réponses approximatives, plutôt qu'exactes. Ces réponses approximatives représentent les graphes les plus similaires au graphe requête. Plusieurs mesures de similarité entre graphes [20, 54, 93, 135, 137] ont été proposées dans la littérature. Chaque mesure a ses propres avantages et inconvénients.

Cette thèse s'inscrit dans le cadre du projet ANR-AOC [1] dont la problématique principale est l'Appariement d'Objets Complexes. Les objectifs de ce projet peuvent se résumer en trois points principaux :

- proposer des méthodes d'appariement de graphes efficaces permettant de prendre en compte la structure topologique et la sémantique associée des graphes comparés,
- traiter les graphes requêtes n'exprimant qu'une partie des besoins de l'utilisateur,
- réduire la complexité des algorithmes d'appariement de graphes, qui sont généralement NP-complets.

Le projet AOC a particulièrement permis d'apporter de nouvelles solutions pour le problème d'appariement de graphes en s'appuyant sur les propriétés structurelles et sur les contraintes sémantiques liées aux graphes à mettre en correspondance. Une seconde contribution majeure de AOC est le traitement des graphes requêtes formulés d'une manière non exhaustive, voir imprécise (due en partie à la méconnaissance des utilisateurs du contenu de la base des graphes cibles). Ce qui permet d'éviter le problème des résultats insatisfaisants (comme, par exemple, la vacuité des réponses) ou inexploitables (comme, par exemple, les réponses pléthoriques qui s'avèrent très difficiles à examiner et à comparer).

## Contributions

L'objectif de cette thèse est ainsi de contribuer à l'interrogation flexible et personnalisée d'objets complexes représentés sous forme de graphes. Plus précisément, nous cherchons à développer un cadre général qui vise à (i) élargir le spectre des approches de traitement de graphes requêtes, d'une part, par la prise en compte des préférences utilisateurs et des contraintes sémantiques sous-jacentes aux modèles de graphes cibles et, d'autre part, par la proposition d'une nouvelle vision de recherche par similarité en exploitant conjointement un ensemble de mesures pour identifier les graphes cibles les

1. Le site officiel du projet est : http ://aoc.irit.fr/.

plus similaires au graphe requête, (ii) améliorer la pertinence des résultats retournés en sélectionnant les plus satisfaisants du point de vue de l'utilisateur.

L'ensemble des travaux présentés dans ce manuscrit se fondent sur des outils théoriques issus de la théorie des ensembles flous.

La première contribution présentée dans cette thèse concerne la recherche de services Web utiles pour une application donnée. Elle est développée dans le cadre du projet ANR-AOC et en collaboration avec l'équipe PRiSM de l'Université de Versailles Saint-Quentin-en-Yvelines. Cette contribution [5, 6, 9, 91, 8, 7] consiste en un système de sélection de services Web modélisés par des graphes (appelés graphes de modèles de processus composés de nœuds connecteurs et de nœuds d'activités) en présence des préférences utilisateurs. Ces préférences sont définies sur les aspects de Qualité de Service (QdS), i.e., sur des aspects non-fonctionnels, comme, par exemple, le coût, le temps de réponse, la disponibilité, etc. Deux types de préférences sont étudiés : (i) des préférences obligatoires (contraintes) dont la satisfaction est obligatoire et (ii) des préférences souples dont la satisfaction n'est pas obligatoire mais plus elles sont satisfaites plus le graphe correspondant est préféré.

Pour valider les propositions visant à améliorer la sélection de services Web en présence des préférences non-fonctionnelles de l'utilisateur, nous avons utilisé la plateforme « *MatchMaker* » de l'équipe PRiSM. Ce qui a permis d'effectuer un ensemble d'expérimentations et de montrer la faisabilité, l'efficacité et la pertinence de l'approche proposée.

Notre deuxième contribution est axée sur la pertinence des résultats retournés lors de l'évaluation d'une requête de recherche de graphes par similarité. Nous avons proposé une approche appelée « *Skyline de graphes par similarité* » [4, 1] qui permet de sélectionner les graphes non-dominés par aucun autre graphe de la base de données, en appliquant la relation de dominance de Pareto. Cette approche a trois avantages principaux : (i) la possibilité d'affranchir l'utilisateur de choisir au hasard une mesure de calcul de distance entre graphes parmi toutes celles proposées dans la littérature, (ii) la préservation des propriétés et des caractéristiques de chacune des mesures de calcul de distances utilisées et enfin (iii) le calcul des graphes les plus similaires au graphe requête qui est l'objectif principal de la requête de recherche de graphes par similarité. Pour valider notre approche et montrer sa faisabilité, un prototype, appelé « *GraphSim* » est développé.

Enfin, notre dernier apport concerne le problème de raffinement de skyline en général. Nous avons défini une nouvelle relation de dominance plus sélective que celle de Pareto [2, 3]. L'objectif de cette relation est de privilégier les éléments du skyline, appelés aussi les éléments ou les points skyline, qui répondent le mieux à la requête de l'utilisateur sur un maximum de dimensions. La solution proposée se fonde sur l'utilisation des quantificateurs linguistiques.

## Organisation du manuscrit

Ce manuscrit est organisé en deux parties.

La première partie est dédiée à l'état de l'art. Cette partie est composée de deux chapitres.

Le chapitre 1 passe en revue les différentes approches de l'interrogation de bases de données de graphes. Dans un premier temps, nous définissons les différentes notions liées aux graphes nécessaires à la lecture de ce manuscrit. Puis, nous décrivons les deux grandes familles d'approches d'évaluation de requêtes à graphes dans le contexte d'une recherche exacte ou approximative de graphes. Une analyse des limites et avantages de ces approches conclut ce chapitre.

Le chapitre 2 présente l'interrogation flexible de bases de données relationnelles où les préférences utilisateur sont prises en compte. Nous discutons les préférences unipolaires et les préférences bipolaires et présentons les différentes approches proposées dans la littérature pour l'expression, l'interprétation et l'évaluation de ces préférences.

Dans la deuxième partie, nous décrivons nos contributions. Cette partie est structurée en quatre chapitres.

Dans le chapitre 3, nous présentons un système de recherche de services Web modélisés sous forme de graphes. Ce système est réalisé dans le cadre du projet ANR-AOC en collaboration avec l'équipe PRiSM, partenaire du même projet. Il permet une interrogation flexible et personnalisée des dépôts de services Web cibles en intégrant les préférences des utilisateurs définies sur les critères de qualité de services Web recherchés. Des mesures ont été intégrées dans ce systèmes pour évaluer la similarité structurelle entre chaque graphe cible du dépôt et le graphe requête ainsi que la satisfaction des préférences de l'utilisateur. Ce qui permet d'établir un ordre entre les services retournés.

Le chapitre 4 discute de l'expérimentation menée pour valider l'approche proposée.

Cette étude s'appuie sur le système « MatchMaker » développé par PRiSM qui a la particularité, contrairement aux systèmes de sélection de services Web, d'introduire des contraintes à différents niveaux de granularité dans la description fonctionnelle des services.

Le chapitre 5 présente une étude sur le skyline par similarité permettant d'évaluer les requêtes à graphes en prenant en considération à la fois plusieurs mesures de calcul de distance entre graphes. Cette approche sélectionne l'ensemble des graphes qui ne sont dominés (au sens de Pareto) par aucun autre graphe de la base interrogée. Nous définissons également dans ce chapitre une méthode de raffinement permettant de réduire la taille de skyline lorsque ce dernier est d'une taille importante.

Dans le chapitre 6, nous nous intéressons au problème de raffinement de skyline en général et proposons trois méthodes de raffinement pour sélectionner les points skyline les plus intéressants vis-à-vis de l'utilisateur. L'idée de base de ces approches est de privilégier les points skyline qui répondent le mieux à la requête de l'utilisateur sur un maximum de dimensions. Le concept clé de ces approches est fondé sur la notion de quantificateur linguistique.

Enfin, un bilan global de la thèse est fourni en conclusion générale où nous rappelons nos différentes contributions. Un ensemble de perspectives est également discuté dans ce chapitre.

# Première partie

# État de l'art

# Chapitre 1

# Sur l'interrogation des bases de données de graphes

## 1.1 Introduction

Les graphes sont devenus de plus en plus importants dans la modélisation des objets complexes et structurés, tels que les composants chimiques, les services Web, les images médicales ou autres, etc. Comme le montrent ces exemples, l'utilisation de graphes ne cesse d'augmenter dans plusieurs domaines d'applications comme la bio-informatique [75, 43, 132], la chimie [139, 82], la reconnaissance d'images [33, 113], les réseaux sociaux [23], la recherche d'information [155, 89], ... Toutes ces applications indiquent ainsi l'importance et l'usage étendu du paradigme des Bases de Données de Graphes (BDG).

L'interrogation de telles bases de données, nécessite des outils informatiques permettant de retrouver le(s) graphe(s) correspondant à la requête de l'utilisateur. Pour ce faire, un processus, dit *processus d'appariement de graphes* est nécessaire pour *évaluer la requête de l'utilisateur, dite requête de recherche de graphes* (ou *requête de graphes*). Ce processus permet ainsi d'identifier et de découvrir les correspondances entre les éléments du graphe de l'utilisateur, appelé *graphe requête*, et ceux de chaque graphe de la BDG, appelés *graphes cibles*. La problématique centrale est la recherche efficace -en termes de temps d'exécution et de qualité des résultats- de graphes cibles correspondant au graphe de la requête. Toutefois, cette efficacité dépend directement de celle du processus d'évaluation de requêtes de recherche de graphes, appliqué sur la base de données interrogée.

Le problème de recherche de graphes peut être scindé en deux catégories : (i) requêtes sous-graphes et ; (ii) requêtes super-graphes. La première catégorie vise à trouver tous les graphes de la base de données qui contiennent le graphe de la requête. Quant à la deuxième catégorie, elle vise à retrouver tous les graphes de la base qui sont contenus dans le graphe requête.

Dans ce chapitre un état de l'art sur l'interrogation des bases de données de graphes est présenté (pour situer le contexte du travail introduit dans ce manuscrit). Dans un premier temps, les différents concepts et notions de bases liés aux graphes, nécessaires à la lecture du manuscrit, sont définis. Aussi, nous décrivons le concept et le principe de l'appariement de graphes, ainsi que les mesures les plus

utilisées pour calculer la distance ou la similarité entre deux graphes. Par la suite, nous présentons les étapes d'évaluation d'une requête de graphe définies dans la littérature. Enfin, nous concluons ce chapitre par une discussion sur la typologie des requêtes de graphes, ainsi que les approches d'interrogation de bases de données de graphes permettant de répondre à ce type de requêtes.

## 1.2 Définitions et notations

Cette section introduit les notions de base sur les graphes utilisées dans le manuscrit.

### 1.2.1 Notion de graphe

Comme structure de données, les graphes sont de plus en plus utilisés pour modéliser des données et objets complexes. Ils permettent de véhiculer un maximum d'informations nécessaires pour assurer une représentation efficace d'objets complexes et aussi une comparaison pertinente entre deux objets (c.-à-d., ces informations ont une sémantique significative permettant de faciliter la comparaison entre deux objets).

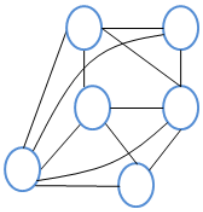
Contrairement aux vecteurs numériques [1], les graphes ont la capacité d'exprimer la structure de chaque objet complexe, telle que la structure d'un composant chimique, d'une image, ou encore d'un document XML. De plus, ils permettent de représenter toutes les relations possibles entres les composants d'un objet, contrairement aux arbres [2] dont les arcs (i.,e. une relation fils-père) se limitent à une relation de sémantique « est de type » ou « appartient à ». En effet, la structure de données d'un graphe, nous permet de véhiculer non seulement les informations sur les composants d'un objet et leurs caractéristiques, mais aussi sur leurs agencements topologiques.

Un objet complexe est modélisé par un graphe dans lequel les nœuds et les arêtes représentent respectivement ses entités (ses composants) et les relations entre ces entités. Les entités et les relations peuvent être étiquetées en attribuant une ou plusieurs valeurs (symboliques ou numériques) à chaque nœud et/ou à chaque arête. Dans ce cas, le graphe est appelé *graphe étiqueté*. Ces étiquettes permettent ainsi de différencier entre les entités et entre les relations d'un objet complexe. Si les nœuds et les arêtes ne possèdent pas d'étiquettes, le graphe est donc appelé *graphe non-étiqueté*. Ce type de graphe est généralement utilisé lorsque seule sa structure ou forme est importante et non pas les étiquettes des nœuds et/ou des arêtes. Néanmoins, tout graphe non-étiqueté peut être représenté par un graphe étiqueté en associant la même étiquette de nœuds et la même étiquette d'arêtes à tous les nœuds et à toutes les arêtes, respectivement.

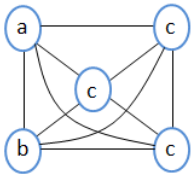
Un graphe (étiqueté ou non) est dit *graphe orienté*, si toutes ses arêtes sont symbolisées par des flèches, sinon il est dit *graphe non-orienté*. À titre d'exemple, les documents XML et les services Web sont représentés par des graphes étiquetés orientés, tandis que les composantes chimiques et réseaux sociaux sont représentés par des graphes étiquetés non-orientés.

1. Un vecteur numérique est composé d'un ensemble de valeurs des attributs et/ou des caractéristiques d'un objet donné.
2. Un arbre est un graphe orienté particulier.

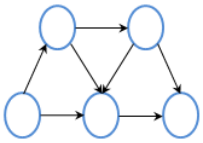
Des exemples de graphes sont illustrés dans la figure 1.1 : (a) un graphe non-étiqueté non-orienté ; (b) un graphe étiqueté non-orienté ; (c) un graphe non-étiqueté orienté et (d) un graphe étiqueté orienté.



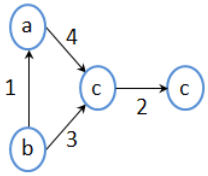
(a) Un graphe non-étiqueté non-orienté.




(b) Un graphe étiqueté non-orienté.



(c) Un graphe non-étiqueté orienté.




(d) Un graphe étiqueté orienté.

FIGURE 1.1 – Exemples de graphes.

La définition formelle d'un graphe est :

**Définition 1.2.1 (Graphe)**. Un **graphe** est défini par un tuple $g = (V, E, L_v, L_e, \nu, \xi)$, où

- $V$ est un ensemble fini de nœuds ;
- $E \subseteq V \times V$ est un ensemble d'arêtes ;
- $L_v$ est un ensemble d'étiquettes de nœuds ;
- $L_e$ est un ensemble d'étiquettes d'arêtes ;
- $\nu$ est une fonction d'étiquetage qui permet d'attribuer à chaque nœud une étiquette dans $L_v$ ;
- $\xi$ est une fonction d'étiquetage qui permet d'attribuer à chaque arête une étiquette dans $L_e$.

Notons aussi qu'un identifiant unique, appelé *id-nœud*, peut être associé à chaque nœud du graphe. Par contre, deux ou plusieurs nœuds peuvent avoir la même étiquette.

Un graphe $g$ est aussi caractérisé par sa taille qui est défini par son nombre de nœuds et/ou son nombre d'arêtes, i.e. $taille(g) = |V|$ ou bien $taille(g) = |E|$ ou bien $taille(g) = |V| + |E|$.

Les différentes définitions d'un sous-graphe sont :

**Définition 1.2.2 (Sous-graphe)**. Soient deux graphes $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$. Le graphe $g$ est un **sous-graphe** de $g'$, noté $g \subseteq g'$ si :

- $V \subseteq V'$ ;

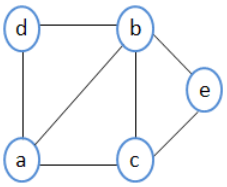
– $E \subseteq E' \cap (V \times V)$ ;
– Pour tout $v$ dans $V$, $\nu(v) = \nu'(v)$ ;
– Pour tout $e$ dans $E$, $\xi(e) = \xi'(e)$.

**Définition 1.2.3 (Sous-graphe induit).** Soient deux graphes $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$. Le graphe $g$ est un **sous-graphe induit** de $g'$, noté $g \subseteq_i g'$, si $E$ est formé de toutes les arêtes de $g'$ ayant leurs extrémités (les deux nœuds formant l'arête) dans $V$, i.e $V \subseteq V'$ et $E = E' \cap V \times V$.

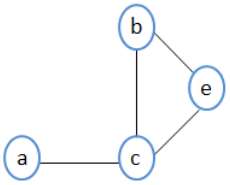
**Définition 1.2.4 (Sous-graphe partiel).** Soient deux graphes $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$. Le graphe $g$ est un **sous-graphe partiel** de $g'$, noté $g \subseteq_p g'$, si $g$ est un graphe qui ne contient pas toutes les arêtes de $g'$ ayant leurs extrémités dans $V$.

Notons que dans les définitions (1.2.2), (1.2.3) et (1.2.4), le graphe $g'$ est dit le **super-graphe** de $g$, noté $g' \supset g$, si $g \subseteq g'$ et $g' \nsubseteq g$.

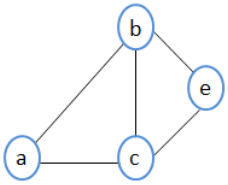
La figure 1.2 montre trois exemples de graphes : (a) un graphe étiqueté non-orienté $g$ ; (b) un sous-graphe partiel de $g$ et (c) un sous-graphe induit de $g$.




(a) Un graphe $g$.




(b) Un sous-graphe partiel de $g$.




(c) Un sous-graphe induit de $g$.

FIGURE 1.2 – Exemple de sous-graphe partiel et induit.

Dans une base de données de graphes, un graphe peut être inclus dans différents graphes de la base et on obtient ainsi les notions de fréquence et de sous-graphe fréquent.

**Définition 1.2.5 (Fréquence / Support d'un graphe).** La *fréquence* (ou *le support*) d'un graphe $g$, notée $freq(g)$ (ou $sup(g)$), est le nombre de graphes dans la base de données $D$ qui le contiennent, c.-à-d., $freq(g) = |D_g|$ (ou $sup(g) = |D_g|$) où $D_g = \{g_i | g_i \in D \wedge g \subseteq g_i\}$.

**Définition 1.2.6 (Sous-graphe fréquent)**. Soit $D = \{g_1, g_2, \ldots, g_n\}$ une base de données qui contient $n$ graphes. Un graphe $g$ est dit *fréquent* si $freq(g) \geq (\sigma.|D|)$, où $\sigma\,(0 < \sigma \leq 1)$ est un paramètre défini par l'utilisateur.

### 1.2.2 Notion de base de données de graphes

Afin d'automatiser la manipulation des graphes, ces derniers sont sauvegardés dans une base de données, appelée **Base de Données de Graphes** (**BDG**). Ainsi, la recherche d'un graphe (objet) donné est réalisée par l'interrogation de cette dernière.

**Définition 1.2.7 (Base de données de graphes).** Une base de données de graphes $D = \{g_1, g_2, ..., g_n\}$ est une collection de graphes $g_i$ $(i = 1, ..., n)$ représentant des objets structurés complexes.

**Définition 1.2.8 (Ensemble de candidats).** Un ensemble de candidats $C_q$ est l'ensemble des graphes de la base de données $D$, qui contiennent toutes les caractéristiques du graphe requête $q$.

À partir de cet ensemble, des tests de vérification sont effectués pour sélectionner les graphes les plus similaires au graphe requête de l'utilisateur en éliminant les fausses réponses.

## 1.3 Appariement de graphes

L'interrogation d'une base de données de graphes a pour objectif d'évaluer une requête de recherche de graphes. Plus précisément, elle a pour objectif :

- de retrouver le ou les graphes les plus similaires au graphe de la requête, comme dans le cas de la recherche d'images [33, 113], de documents XML [155, 89] ou de services Web [57] ;
- de retrouver le ou les graphes contenant le graphe de la requête comme dans le cas de l'analyse du comportement d'un composant dans le domaine de biologie [132] ;
- d'identifier le ou les graphes contenus dans le graphe de la requête comme dans le cas de l'étude des propriétés d'un nouveau composant chimique plus complexe [134].

Pour comparer deux graphes, *un processus d'appariement* est donc utilisé pour effectuer une mise en correspondance de leurs nœuds et/ou de leurs arêtes. Si une mise en correspondance de tous les nœuds, de toutes les arêtes et des fonctions d'étiquetage est déterminée, alors les deux graphes sont strictement identiques et dits **isomorphes** sinon ils sont différents. Deux types de processus d'appariement peuvent être distingués : (i) l'*appariement exact* de graphes qui impose des contraintes obligatoires sur le processus de mise en correspondance et ; (ii) l'*appariement approximatif* de graphes qui utilise des contraintes plus souples pour permettre une mise en correspondance avec une certaine tolérance d'erreurs. Ces deux catégories d'appariement de graphes sont introduites dans les sections 1.3.1 et 1.3.2, respectivement.

### 1.3.1 Appariement exact de graphes

L'objectif d'un appariement exact est de déterminer si deux graphes, ou des parties de deux graphes sont identiques suivant leurs structures et leurs étiquettes. Cette égalité entre deux graphes est alors définie par une fonction bijective, dite **isomorphisme de graphe**.

**Définition 1.3.1** (**Isomorphisme de graphe**). Soient deux graphes $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$. Un isomorphisme de graphe entre $g$ et $g'$ est une *fonction bijective* $f : V \rightarrow V$ ' avec $|V| = |V'|$, où

- pour tout nœud $v \in V$, il existe un nœud $v' = f(v) \in V'$ tel que $\nu(v) = \nu'(f(v))$,
- pour toute arête $e = (u, v) \in E$, il existe une arrête $e' = (f(u), f(v)) \in E'$ telle que $\xi(e) = \xi'(e')$,
- pour toute arête $e' = (u', v') \in E'$, il existe une arrête $e = (f^{-1}(u'), f^{-1}(v')) \in E'$ telle que $\xi(e) = \xi'(e')$.

Ainsi, le graphe $g$ est dit *isomorphe* au graphe $g'$, s'il existe un isomorphisme entre les deux graphes.

L'appariement exact de graphes consiste donc à déterminer une égalité entre deux graphes en terme de structure et d'étiquettes. Il permet ainsi de retrouver une mise en correspondance entre tous les nœuds et/ou arêtes du graphe requête et du graphe cible tout en respectant les fonctions d'étiquetage. La figure 1.3 donne un exemple de deux graphes isomorphes en appliquant une fonction bijective $f$ telle que : $f(a) = 1$, $f(b) = 2$, $f(c) = 3$, $f(d) = 4$ et $f(e) = 5$.


(a) Un graphe $g$. (b) Un graphe $g'$.

FIGURE 1.3 – Exemple d'isomorphisme de graphe.

Dans certains cas, la recherche des objets contenant un autre objet peut être plus intéressant. En effet, en chimie organique, par exemple, les chimistes ont parfois besoin de trouver toutes les molécules complexes contenant une autre molécule afin d'étudier ses comportements et ses propriétés chimiques. En recherche de services Web, il est parfois nécessaire de rechercher un service particulier contenu dans d'autres services Web. En reconnaissance d'images, la recherche peut être réalisée en récupérant toutes les images qui contiennent un objet particulier.

Le problème des cas cités ci-dessus consiste à prouver si le graphe de plus petite taille est bien contenu dans le graphe de plus grande taille. Pour ce faire, le processus d'*appariement de sous-graphes* est appliqué. Cet appariement permet de retrouver une partie du graphe cible identique au graphe de la requête. Autrement dit, le problème est ramené à celui de trouver un **isomorphisme de sous-graphe** entre le graphe requête et le graphe cible.

**Définition 1.3.2 (Isomorphisme de sous-graphe).** Soient deux graphes $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$. Un isomorphisme de sous-graphe entre $g$ et $g'$ est une *fonction injective* $f : V \rightarrow V'$ avec $|V| \leq |V'|$, telle que

- pour tout nœud $v \in V$, il existe un nœud $v' = f(v) \in V'$ tel que $\nu(v) = \nu'(f(v))$,
- pour toute arête $e = (u, v) \in E$, il existe une arrête $e' = (f(u), f(v)) \in E'$ telle que $\xi(e) = \xi'(e')$.

Un isomorphisme de sous-graphe existe entre deux graphes $g$ et $g'$, s'il existe un sous-graphe $g''$ de $g'$ (i.e. $g'' \subseteq g'$) tel que $g''$ et $g$ sont isomorphes de graphe. Autrement dit, l'isomorphisme de sous-graphe entre deux graphes indique que le graphe de plus petite taille est inclus dans le graphe de plus grande taille.

Un exemple d'isomorphisme de sous-graphe est illustré dans la figure 1.4, en appliquant une fonction injective $f$ telle que : $f(a) = 1$, $f(b) = 2$, $f(c) = 3$, $f(d) = 4$ et $f(e) = 5$.


(a) Un graphe $g$. (b) Un graphe $g'$.

FIGURE 1.4 – Exemple d'isomorphisme de sous-graphe.

Le problème d'isomorphisme de graphes n'est pas encore déterminé s'il est dans la classe P ou dans la classe NP-complet [56]. Cependant, il a été montré que la complexité de ce problème peut être polynomiale pour certains types de graphes, tels que les graphes planaires [3] [74] et les arbres [112].

Les notions de sous-graphes, de requêtes de graphes et d'appariement exact entre graphes sont définies par :

**Définition 1.3.3 (Sous-graphe v.s. super-graphe).** Un graphe $g$ est un *sous-graphe* d'un autre graphe $g'$ (resp., $g'$ est un *super-graphe* de $g$), noté $g \subseteq g'$ (resp., $g' \supseteq g$), s'il existe un isomorphisme de sous-graphe entre $g$ et $g'$.

**Définition 1.3.4 (Problème de requêtes sous-graphes).** Soient une base de données $D = \{g_1, ..., g_n\}$ et un graphe requête $q$. Le problème de requête sous-graphe est de trouver tous les graphes $g_i \in D$ tel que $q \subseteq g_i$. Soit $D_q$ l'ensemble de réponses de $q$, on a $D_q = \{g_i | g_i \in D \wedge q \subseteq g_i\}$.

3. Les graphes planaires sont des graphes qui peuvent être redessinés sans que leurs arêtes ne se croisent.

**Définition 1.3.5 (Problème de requêtes super-graphes).** Soient une base de données $D = \{g_1, ..., g_n\}$ et un graphe requête $q$. Le problème de requête super-graphe est de trouver tous les graphes $g_i \in D$ tel que $q \supseteq g_i$. Soit $D_q$ l'ensemble de réponses de $q$, on a $D_q = \{g_i | g_i \in D \wedge g_i \subseteq q\}$.

**Définition 1.3.6 (Appariement exact de graphes / sous-graphes).** Soient deux graphes $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$. *Un appariement exact de graphes* est une fonction de mise en correspondance bijective $f : V \rightarrow V'$, où pour chaque $v \in V$, $l(v) = l'(f(v))$, pour chaque $(u, v) \in E$, $(f(u), f(v)) \in E'$, et pour chaque $(u', v') \in E'$, $e = (f^{-1}(u'), f^{-1}(v')) \in E$ et $\xi(e) = \xi'(e')$.

D'une manière similaire, *un appariement exact de sous-graphes* est une fonction de mise en correspondance injective $f' : V \rightarrow V'$, où pour chaque $v \in V$, $\nu(v) = \nu'(f(u))$ et pour chaque $(u, v) \in E$, $(f(u), f(v)) \in E'$. En d'autres termes, l'appariement exact de sous-graphes entre $g$ et $g'$ est défini comme suit : $\exists g'', g'' \subseteq g'$, et $g''$ est isomorphe au graphe $g$.

L'appariement exact de graphes entre $g$ et $g'$, permet alors de vérifier si les deux graphes sont isomorphes. Quant à l'appariement exact de sous-graphes, il vérifie s'il existe un sous-graphe $g$" dans $g'$ qui est isomorphe au graphe $g$.

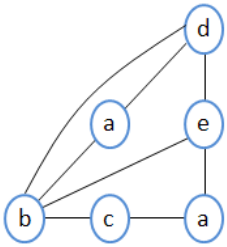
### 1.3.2 Appariement approximatif de graphes

L'appariement exact de graphes (ou de sous-graphes) peut facilement induire aux réponses vides lorsqu'aucun graphe de la base de données n'est pas exactement égal au (ou ne contient le) graphe de la requête. Cependant, en pratique, deux graphes sont rarement identiques, même s'ils ont été extraits du même objet. En effet, des différences structurelles peuvent être produites par l'incertitude lors du processus d'extraction de caractéristiques, par des environnements bruités, etc. Pour cela, un appariement approximatif entre graphes est nécessaire pour identifier la correspondance entre deux graphes malgré l'existence de différences structurelles mineures.

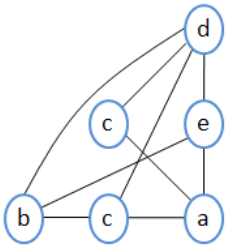
**Définition 1.3.7 (Appariement approximatif de graphes / sous-graphes).** *L'appariement approximatif de graphes* tolère des dissemblances entre un certain nombre de nœuds et/ou d'arêtes (c.-à-d., $l(v) \neq l'(f(v))$), ainsi que des suppressions et des insertions de nœuds et/ou d'arêtes des deux graphes comparés. Autrement dit, l'appariement approximatif de graphes entre $g = (V, E, L_v, L_e, \nu, \xi)$ et $g' = (V', E', L'_v, L'_e, \nu', \xi')$ est une fonction de correspondance bijective $f : V_1 \rightarrow V_2$, où $V_1 \subseteq V$et $V_2 \subseteq V'$ (voir la définition (1.3.6)).

D'une manière similaire, *un appariement approximatif de sous-graphes* entre $g$ et $g'$ est une fonction de correspondance injective $f' : V_1 \rightarrow V_2$, où $V_1 \subseteq V$ et $V_2 \subseteq V'$. En d'autre termes, l'appariement approximatif de sous-graphes entre $g$ et $g'$ est défini comme suit : $\exists g''$, $g'' \subseteq g'$ et $g$ est le graphe correspondant approximativement au graphe $g''$.

L'objectif d'un appariement approximatif de graphes est donc de déterminer une correspondance entre deux graphes avec certaines tolérances. Considérons les deux graphes $g$ et $g'$ de la figure 1.5. Il est clair que ces deux graphes sont très similaires, au regard de leurs nœuds et arêtes qui sont identiques. Néanmoins, ces graphes ne sont pas isomorphes de (sous-)graphes. Ainsi, lors de l'application d'un appariement exact, les graphes sont considérés différents, tandis que le processus d'appariement approximatif de (sous-)graphes considère ces graphes similaires après avoir évaluer leur ressemblance avec tolérance de certaines de leurs différences.




(a) Un graphe $g$.




(b) Un graphe $g'$.

FIGURE 1.5 – Exemple de graphes similaires.

La comparaison de deux graphes à l'aide d'un processus d'appariement approximatif de graphes ou de sous-graphes, nécessite une méthode de calcul de similarité ou de distance entre graphes. Dans la littérature, plusieurs méthodes ont été proposées. Dans la sous-section suivante, nous présentons les mesures les plus utilisées pour calculer la similarité entre deux graphes et discutons les caractéristiques de chacune.

### 1.3.3 Mesures de similarité entre graphes

Afin de comparer deux graphes lors d'un appariement exact ou approximatif de graphes ou de sous-graphes, il est nécessaire de disposer d'une mesure permettant de calculer la similarité ou la distance entre deux graphes. Une telle mesure doit être capable de quantifier les caractéristiques communes ainsi que les différences entre deux graphes. La comparaison de deux graphes consiste en la comparaison de leur structure tout en prenant en considération les étiquettes des nœuds et des arêtes ainsi que leurs fonctions d'étiquetage.

Un graphe est une structure complexe et comprend une multitude de caractéristiques de base. Ainsi, il est difficile de donner une définition significative de la similarité entre graphes. Plusieurs modèles [17, 20, 137] ont été proposés pour mesurer la similarité ou la distance entre deux graphes. À l'aide de ces mesures, on peut évaluer deux graphes et dire s'ils sont identiques ou similaires ou complètement différents. Plus exactement, ces mesures évaluent le degré de similitude structurelle de deux graphes.

#### 1.3.3.1 Distance d'édition de graphes

La distance d'édition a été à la base proposée par Levenstein [93] pour calculer la distance entre deux chaînes de caractères. Ensuite, cette mesure a été généralisée et adaptée aux arbres [117, 128] puis aux graphes [135, 51]. L'idée de base de la distance d'édition est de mesurer une dissimilarité en déterminant l'ensemble le moins coûteux des opérations nécessaires pour transformer un ensemble (chaîne de caractères) en un autre.

La distance d'édition de graphes a donc pour objectif de déterminer l'ensemble de coût minimal des opérations nécessaires pour transformer un graphe en un autre. Pour ce faire, un ensemble d'opérations d'édition est défini et constitué de : (i) suppression de nœuds ou d'arêtes, (ii) insertion de nœuds ou d'arêtes et, (iii) substitution des étiquettes des nœuds ou des arêtes. Ensuite, un coût de pénalité (un nombre réel non négatif) est attribué à chaque opération d'édition reflétant l'intensité de distorsion induite par la transformation.

Soient $e_i$ une opération d'édition et $c(e_i)$ son coût. Le coût d'une séquence d'opérations d'édition, notée $s = (e_1, ..., e_n)$ est donné par :

$$c(s) = \sum_{i=1}^{n} c(e_i) \tag{1.1}$$

En se basant sur cette fonction de coûts $c$, la distance d'édition de graphes est définie comme suit.

**Définition 1.3.8 (Distance d'édition de graphes).** La distance d'édition entre deux graphes $g$ et $g'$ est égale au coût minimal résultant de toutes les séquences d'opérations possibles qui transforment $g$ en $g'$, i.e.,

$$Dist_{Ed}(g, g') = \min_{s \in E} c(s) \tag{1.2}$$

où $E$ dénote l'ensemble de toutes les séquences d'opérations d'édition possibles pour transformer $g$ en $g'$. Plus $Dist_{Ed}(g, g')$ est faible, plus les deux graphes sont similaires.

La fonction de coût joue un rôle très important dans la distance d'édition de graphes. Elle permet d'adapter les coûts des distorsions causées en pénalisant plus ou moins les opérations d'édition à appliquer. Néanmoins, il est difficile de déterminer le coût de chaque opération d'édition élémentaire [18]. À cet effet, plusieurs travaux ont été réalisés dans l'objectif de faciliter le paramétrage de fonctions des coûts [107, 108].

Plusieurs séquences d'opérations d'édition peuvent exister pour transformer un graphe en un autre. Un exemple de séquences d'opérations d'édition est présenté dans la figure 1.6, permettant de transformer le graphe $g$ en graphe $g'$ de la figure 1.5 page précédente. La transformation est donc composée de quatre opérations d'édition : (i) la substitution du nœud $a$ par $c$ ; (ii) la suppression de l'arête $(b, c)$ et ; (iii) deux insertions d'arêtes : $(c, a)$ et $(c, d)$. En utilisant une distance d'édition uniformément pondérée où chaque opération d'édition possède le même coût (égale à 1), on peut

vérifier que cette séquence est la minimale (i.e., la meilleure). La distance d'édition entre les deux graphes $g$ et $g'$ est ainsi $Dist_{Ed}(g, g') = 3$.

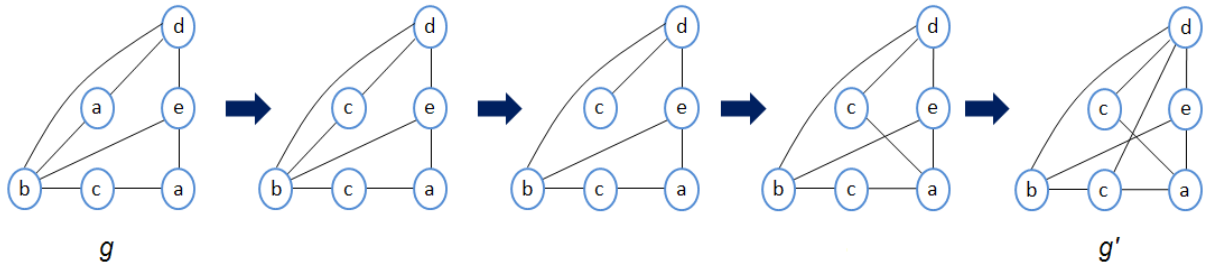



FIGURE 1.6 – Exemple de séquences d'opérations d'édition pour transformer $g$ en $g'$.

Afin de calculer la distance d'édition entre deux graphes, plusieurs algorithmes ont été proposés [55, 52]. Ces derniers peuvent être divisés en deux catégories : les algorithmes dits optimaux et, les algorithmes approximatifs. Les algorithmes optimaux [21, 116] sont basés sur la construction automatique d'un arbre de recherche qui contient tous les appariements possibles entre les nœuds et les arêtes du premier graphe et les nœuds et les arêtes du deuxième graphe. Toutefois, la complexité de ces algorithmes varie d'une façon exponentielle en fonction du nombre de nœuds des graphes appariés. En conséquence, l'application de la distance d'édition est restreinte, en pratique, aux graphes de petite taille.

Des algorithmes approximatifs [78, 109, 115, 154, 52] calculant une distance approchée, ont ensuite été proposés pour réduire la complexité des calculs de la distance d'édition de graphes. Dans [109], deux algorithmes sous-optimaux permettant de calculer une approximation de la distance d'édition (distance approchée) entre deux graphes d'une façon très efficace, ont été proposés. Dans [52], les auteurs proposent un nouvel algorithme de calcul de distance d'édition approchée en se basant sur une procédure efficace d'optimisation bipartite sous-optimale. Dans [154], trois méthodes ont été développées pour calculer en un temps polynomial les deux bornes, inférieure et supérieure, de la distance d'édition entre deux graphes.

**Les points forts et les points faibles de la distance d'édition :**

La distance d'édition est une mesure très utilisée pour évaluer le degré de dissimilarité de deux graphes. Ceci est en dû à trois raisons principales. Tout d'abord, la mesure étant intuitive, l'utilisateur peut aisément comprendre comment est calculée la distance entre deux graphes. De plus, les paramètres de la fonction de coûts peuvent être adaptés et modifiés en fonction des besoins des utilisateurs. Enfin, sa grande capacité d'adaptation aux différentes applications lui permet d'être utilisée sur une grande variété de graphes.

Toutefois, cette méthode souffre de certains inconvénients, notamment la difficulté de déterminer les coûts de chaque opération d'édition, et ce malgré la possibilité de paramétrer la fonction des coûts suivant les attentes de l'utilisateur. De plus, la complexité exponentielle de son algorithme réduit son utilisation aux petits graphes.

#### 1.3.3.2 Distance basée sur le sous-graphe commun maximal

Un nouveau concept, dit *sous-graphe commun maximal*, noté *scm*, a été introduit pour évaluer la similarité entre deux graphes. Ce dernier permet de calculer une similarité partielle entre deux graphes.

**Définition 1.3.9 (Sous-graphe commun maximal).** Soient deux graphes $g$ et $g'$. Un graphe $g''$ est dit un *sous-graphe commun* de $g$ et $g'$, s'il existe un isomorphisme de sous-graphe entre $g''$ et $g$ et entre $g''$ et $g'$. Un sous-graphe commun $g''$ de $g$ et $g'$ est dit *maximal* si et seulement s'il n'existe pas un autre sous-graphe commun de $g$ et $g'$ ayant un nombre de nœuds et/ou d'arêtes supérieur à celui de $g''$.

Donc, plus le sous-graphe commun maximal de deux graphes est grand (relativement à la taille des graphes), plus les graphes sont similaires. Le sous-graphe commun maximal peut être vu comme l'intersection entre deux graphes, définissant la plus grande partie commune en terme d'étiquettes et de topologie (structure). La figure 1.7 montre le sous-graphe commun maximal entre les graphes $g$ et $g'$ de la figure 1.5 page 17.

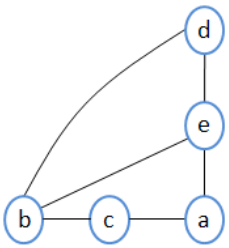



FIGURE 1.7 – Sous-graphe commun maximal des deux graphes $g$ et $g'$.

En se basant sur la notion de sous-graphe commun maximal, Bunke et Shearer [20] ont développé une mesure de calcul de distance entre deux graphes.

**Définition 1.3.10 (Distance basée sur le sous-graphe commun maximal).** Soient deux graphes $g$ et $g'$. La distance basée sur le sous-graphe commun maximal entre $g$ et $g'$ est définie comme suit :

$$Dist_{scm}(g, g') = 1 - \frac{|scm(g, g')|}{max(|g|, |g'|)} \tag{1.3}$$

où $|scm(g, g')|$ dénote la taille du sous-graphe commun maximal de $g$ et $g'$ (c.-à-d., le nombre de nœuds et/ou d'arêtes).

La distance $Dist_{scm}$ est normalisée, i.e. $0 \leq Dist_{scm}(g, g') \leq 1$. En effet, si deux graphes sont très similaires alors leur sous-graphe commun maximal leur est très semblable. Par conséquent, la taille du sous-graphe commun maximal est très proche de celle de l'un des deux graphes, ainsi la fraction

tend vers 1 et la distance tend vers 0. Inversement, si deux graphes comparés sont très dissimilaires, alors la taille de leur sous-graphe commun maximal est très petite. La fraction tend dans ce cas vers 0 et la distance quant à elle tend vers 1.

Supposons que la taille d'un graphe donné est le nombre d'arêtes le composant. À partir des deux graphes présentés dans la figure 1.5 page 17, la distance basée sur le sous-graphe commun maximal de $g$ et $g'$ est $Dist_{scm}(g, g') = 0,33$ car $|scm(g, g')| = 6$, $max(|g|, |g'|) = |g| = 9$ et $|g'| = 8$.

Dans le cas où la taille d'un graphe est définie comme son nombre de nœuds, Bunke et Shearer [20] ont démontré que la distance basée sur le sous-graphe commun maximal est une métrique vérifiant les propriétés suivantes, quelques soient les graphes $g$, $g'$ et $g''$ :

- $0 \leq Dist_{scm}(g, g') \leq 1$ ;
- $Dist_{scm}(g, g') = 0 \Leftrightarrow g$ et $g'$ sont isomorphes ;
- $Dist_{scm}(g, g') = Dist(g', g)$ (symétrie) ;
- $Dist_{scm}(g, g') + Dist_{scm}(g', g'') \geq Dist_{scm}(g, g'')$ (inégalité triangulaire).

La mesure de calcul de similarité entre graphes, $Sim_{scm}$, dérivée de $Dist_{scm}$ peut s'écrire comme suit :

$$Sim_{scm} = \frac{|scm(g, g')|}{max(|g|, |g'|)}$$

L'avantage principal de l'approche basée sur le sous-graphe commun maximal est la non-utilisation de fonctions de coût, palliant ainsi un inconvénient de l'approche basée sur la distance d'édition. Néanmoins, cette approche dépend fortement de l'efficacité de l'algorithme utilisé pour déterminer le sous-graphe commun maximal.

Plusieurs méthodes de calcul de sous-graphe commun maximal ont été proposées [138, 85, 104]. Cependant, la complexité de leurs algorithmes est exponentielle au nombre de nœuds des deux graphes comparés. Dans [19, 35, 34], quelques unes de ces méthodes ont été comparées sur de grandes bases de données de graphes. Un algorithme avec une complexité de $O(2^n)$ a aussi été présenté par Shearer et Bunke dans [121]. Par conséquent, des algorithmes approximatifs ont été développés [114].

**Les points forts et les points faibles de la distance basée sur le sous-graphe commun maximal :**

Les deux points forts de la distance basée sur le sous-graphe commun maximal, résident dans le fait qu'aucune fonction de coût n'est nécessaire pour calculer la distance entre deux graphes et que son application n'est pas restreinte à un certain type de graphes.

Dans [17], les auteurs ont montré qu'il existe une forte relation entre la distance d'édition et la taille du sous-graphe commun maximal. Toutefois, les deux problèmes restent NP-difficiles [56]. Enfin, la taille du plus petit graphe n'est pas prise en compte dans le calcul de la distance entre deux graphes de différentes tailles, car en considérant trois graphes $g$, $g'$ et $g$" tel que $|g"| > |g'| > |g|$ et $|scm(g, g")| = |scm(g', g")|$, la distance basée sur le sous-graphe commun maximal entre $g$ et $g$" et celle entre $g'$ et $g$" sont de même valeur (i.e., $Dist_{scm}(g, g") = Dist_{scm}(g', g")$) malgré les

différences structurelles entre le graphe $g$ et le graphe $g'$. C'est pourquoi la distance présentée dans la sous-section suivante (i.e., la distance basée sur l'union de graphes) a été proposée.

#### 1.3.3.3 Distance basée sur l'union de graphes

Dans [137], Wallis *et al.* proposent une autre mesure de calcul de distance entre graphes, en se basant sur le principe de l'union de graphes au lieu de la taille du plus grand graphe.

**Définition 1.3.11 (Distance basée sur l'union de graphes).** Soient deux graphes $g$ et $g'$.

La distance basée sur l'union de graphes entre $g$ et $g'$ est définie comme suit :

$$Dist_{ug}(g, g') = 1 - \frac{|scm(g, g')|}{|g| + |g'| - |scm(g, g')|} \tag{1.4}$$

où le dénominateur représente la taille de l'union des deux graphes $g$ et $g'$ selon une vue ensembliste [4].

Cette mesure de distance basée sur l'union de graphes est aussi normalisée, i.e. $0 \leq Dist_{ug}(g, g') \leq 1$. En effet, si les deux graphes $g$ et $g'$ sont très similaires alors leur sous-graphe commun maximal leur est très semblable, i.e., $|g| \simeq |g'| \simeq |scm|$. Par conséquent, le dénominateur représentant la taille de l'union des deux graphes $g$ et $g'$ tend vers la taille du sous-graphe commun maximal, i.e., $|g| + |g'| - |scm(g, g')| \simeq |scm(g, g')|$. Ainsi, la fraction tend vers 1 et la distance tend vers 0. Inversement, si deux graphes comparés sont très dissimilaires, alors la taille de leur sous-graphe commun maximal est très petite et la taille de l'union des deux graphes $g$ et $g'$ est très grande. Ainsi, la fraction tend vers 0 et la distance tend vers 1.

Supposons que la taille d'un graphe est déterminée par le nombre d'arêtes le constituant. Ainsi, la distance basée sur l'union des deux graphes $g$ et $g'$ présentés dans la figure 1.5 page 17, est $Dist_{ug}(g, g') = 0.45$.

Comme pour la mesure basée sur le sous-graphe commun maximal, la distance basée sur l'union de graphes est une métrique et son comportement est assez proche de celui de $Dist_{scm}$. Toutefois, la mesure basée sur l'union de graphes est plus exigeante que celle basée sur le sous-graphe commun maximal. En effet, à partir de leurs formules de calcul de distance entre graphes, il est facile de voir que $Dist_{ug}(g, g') \geq Dist_{scm}(g, g')$.

La mesure de similarité de graphes dérivée de $Dist_{ug}$ s'écrit comme suit :

$$Sim_{ug}(g, g') = \frac{|scm(g, g')|}{|g| + |g'| - |scm(g, g')|}$$

L'utilisation du concept d'union de graphes [137] est motivée par le fait que *les changements dans la taille du plus petit graphe* qui préservent le sous-graphe commun maximal constant entre deux graphes, ne sont pas pris en considération dans la mesure basée sur le sous-graphe commun

4. Cette mesure de distance ressemble à l'indice de Jaccard utilisé pour mesurer la similarité entre deux ensembles $A$ et $B$, i.e., $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

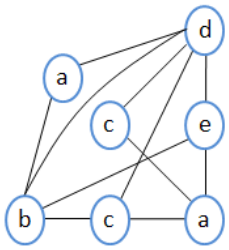



FIGURE 1.8 – Super-graphe commun minimal des deux graphes $g$ et $g'$.

maximal. Tandis que la mesure basée sur l'union de graphes prend en compte cette variation.

**Les points forts et les points faibles de la distance basée sur l'union de graphes :**

Le point fort de la mesure basée sur l'union de graphes réside dans sa prise en considération de la taille des deux graphes comparés. De plus, l'union de deux graphes $g$ et $g'$ contient non seulement les informations communes mais aussi les différences entres eux. Néanmoins, elle souffre de la complexité exponentielle des algorithmes calculant le sous-graphe commun maximal de deux graphes.

#### 1.3.3.4 Distance basée sur le super-graphe commun minimal

Afin de calculer la distance entre deux graphes, Fernández et Valiente [54] proposent une autre mesure basée sur la notion de *sous-graphe commun maximal* et de *super-graphe commun minimal*.

**Définition 1.3.12 (Super-graphe commun minimal).** Soient deux graphes $g$ et $g'$. Un graphe $g''$ est dit *super-graphe commun* des deux graphes $g$ et $g'$ s'il existe un isomorphisme de sous-graphe entre $g$ et $g''$ et entre $g'$ et $g''$. Un super-graphe commun $g''$ des deux graphes $g$ et $g'$ est *minimal* si et seulement s'il n'existe pas un autre super-graphe commun de $g$ et $g'$ ayant un nombre de nœuds et/ou d'arêtes inférieur à celui de $g''$.

La figure 1.8 illustre le super-graphe commun minimal des deux graphes $g$ et $g'$ présentés dans la figure 1.5 page 17.

En se basant sur la notion de super-graphe commun minimal, noté $SCM$, la distance entre graphes développée dans [54] est la suivante :

**Définition 1.3.13 (Distance basée sur le super-graphe commun minimal).** Soient deux graphes $g$ et $g'$. La distance basée sur le super-graphe commun minimal entre $g$ et $g'$ est définie comme suit :

$$Dist_{SCM}(g, g') = 1 - \frac{|scm(g, g')|}{|SCM(g, g')|} \tag{1.5}$$

où le dénominateur représente la taille du super-graphe commun minimal des deux graphes $g$ et $g'$.

La distance $Dist_{SCM}$ est normalisée et son comportement est très similaire à celui de la distance $Dist_{ug}$. Comme dans la distance $Dist_{ug}$, la mesure $Dist_{SCM}$ prend en considération d'une manière indirecte la taille des deux graphes comparés. En effet, le super-graphe commun minimal des deux graphes $g$ et $g'$ est un graphe $g''$ avec une structure minimale mais nécessaire pour contenir les deux graphes $g$ et $g'$. Ainsi, il contient les informations communes et les différences existantes entre les deux graphes comparés $g$ et $g'$.

La mesure de similarité de graphes dérivée de $Dist_{SCM}$ s'écrit :

$$Sim_{SCM}(g, g') = \frac{|scm(g, g')|}{|SCM(g, g')|}$$

L'idée de base de la distance $Dist_{SCM}$ est que pour deux graphes similaires, la taille de leur sous-graphe commun maximal et celle de leur super-graphe commun minimal doivent être très similaires. La distance est ainsi minimale.

Reprenons les deux graphes $g$ et $g'$ illustrés dans la figure 1.5 page 17. En considérant que la taille d'un graphe est le nombre de ses arêtes, la distance basée sur le super-graphe commun minimal de $g$ et $g'$ est $Dist_{SCM}(g, g') = 0.45$.

**Les points forts et les points faibles de la distance basée sur le super-graphe commun minimal :**

Comme pour la mesure, $Dist_{ug}$, basée sur l'union de graphes, la distance, $Dist_{SCM}$, basée sur le super-graphe commun minimal prend en compte la taille des deux graphes comparés. Par conséquent, il n'y a aucune perte d'information lors du calcul de la distance entre deux graphes.

Il a été montré dans [22] que le calcul du super-graphe commun minimal de deux graphes peut être effectué en évaluant leur sous-graphe commun maximal. Par conséquent, les algorithmes calculant le sous-graphe commun peuvent être utilisés pour calculer le super-graphe commun minimal. Toutefois, le problème de calcul de super-graphe commun minimal est NP-complet.

## 1.4 Étapes d'interrogation de bases de données de graphes

D'une manière générale, les approches d'interrogation de bases de données de graphes pour évaluer une requête utilisateur adoptent une démarche composée des trois phases suivantes :

1. Indexation

   Dans le but de réduire l'espace de recherche, la plupart des approches d'évaluation de requêtes de graphes construisent tout d'abord leurs index (voir, par exemple, GIndex dans [145]) sur l'ensemble des caractéristiques des graphes de la base de données. Ces caractéristiques sont extraites au moyen de méthodes spécifiques telle la FSG [88], la CloseGraph [144] ou encore la QuickSI [118].

2. Filtrage

La phase de filtrage permet d'obtenir l'ensemble des réponses candidates à la requête, en sélectionnant à partir de l'index les graphes de la base de données ayant les mêmes caractéristiques que celles du graphe de la requête. Certaines approches tolèrent l'absence de quelques caractéristiques de la requête dans les graphes de la base de données, par exemple, Grafil [146].

3. Vérification
   Cette dernière phase permet d'élaguer les fausses réponses en effectuant un ensemble de tests d'isomorphisme entre chaque graphe candidat et le graphe de la requête.

Le gain est d'autant plus grand que l'ensemble des graphes candidats est de taille réduite par rapport à la taille de la base interrogée ; ainsi le filtrage est plus rapide.

Afin de réduire l'espace de recherche et rendre plus efficace la recherche de graphes, de nombreuses méthodes utilisant des techniques d'indexation ont été développées. Le tableau 1.1 résume les unités d'indexation les plus utilisées dans la littérature.

## 1.5 Typologie de requêtes de graphes

Le problème de requêtes de recherche de graphes dans une base de données de graphes peut être scindé en deux sous-problèmes : (i) *la recherche de requêtes sous-graphes fondée sur une relation d'inclusion* ; et (ii) *la recherche de requêtes super-graphes fondée sur une relation d'exclusion.*

Soient $D = \{g_1, g_2, ..., g_n\}$ une base de données de $n$ graphes et $q$ une requête de graphes.

1. *La recherche de requêtes sous-graphes :* il s'agit de trouver tous les graphes $g_i \in D$ telle que la requête $q$, dite dans ce cas ***requête sous-graphe***, est incluse dans $g_i$ (i.e. $q \subseteq g_i$) ;
2. *La recherche de requêtes super-graphes :* il s'agit de rechercher tous les graphes $g_i \in D$ tel que $g_i$ est inclus dans la requête $q$ (i.e. $g_i \subseteq q$). Dans ce cas, la requête est appelée ***requête super-graphe***.

Remarquons qu'une requête sous-graphe ou super-graphe se ramène au même problème d'un appariement de sous-graphe.

Il a été montré dans la littérature [28, 119, 134, 156] que l'opération d'appariement est centrale dans le processus d'évaluation de requêtes de recherche de graphes. Ainsi, on distingue deux familles d'approches d'évaluation. La première famille est basée sur l'appariement exact de graphes ou de sous-graphes entre le graphe de la requête et les graphes de la base de données (la recherche exacte de graphes). La deuxième famille est fondée sur l'appariement approximatif de graphes ou de sous-graphes entre les graphes de la base de données et le graphe de la requête (la recherche approximative ou par similarité de graphes). Cette dernière famille permet ainsi de pallier le problème de réponses vides que peut produire la famille de recherche exacte de graphes ou de sous-graphes.

La recherche de graphes par similarité qui consiste à rechercher tous les graphes d'une base de données de graphes similaires au graphe de la requête, est apparue comme la nouvelle tendance pour les raisons suivantes. Premièrement, de nombreuses et réelles bases de données de graphes

TABLE 1.1 – Exemples d'unités d'indexation de graphes.

| **Unité d'indexation** | **Description de la technique** | **Exemples d'approches** |
|---|---|---|
| Chemin | – Énumération de tous les chemins ayant une certaine longueur dans un graphe donné. | – GraphGrep [60]. |
| Sous-graphe fréquent / fragment fréquent [5] | – Calcul de tous les sous-graphes de la base $D$, ayant une fréquence supérieure ou égale à un certain seuil (i.e. supérieure à $\sigma.\|D\|$, voir la définition (1.2.6) page 5). | – GIndex [145]. |
| Séquence | – Transformation des graphes en chaînes de caractères pour effectuer une comparaison lexicographique entre deux graphes. | – GString [77]. |
| Arbres | – Représentation des graphes de la base de données sous forme d'arbres en utilisant certaines notions, comme la fermeture de graphes [71] ou les sous-graphes communs [157]. | – C-Tree [71].<br>– GPTree [157]. |
| Nœuds importants | – Utilisation des nœuds ayant un nombre élevé de nœuds voisins. | – VFM [94].<br>– Tale [133]. |

sont de nature bruitée et incomplète (par exemple, les graphes représentant des images médicales peuvent être imparfaits dû au dysfonctionnement des scanners, mauvais réglage d'appareil, etc.) [122, 124], d'où la nécessité d'un *appariement approximatif* de graphes. Deuxièmement, plusieurs applications modernes (telles que la médecine, l'imagerie, ...) préfèrent les résultats d'un appariement approximatif plutôt que ceux d'un appariement exact, car ils véhiculent davantage d'informations, comme ce qui pourrait être manquant ou superflu dans un graphe de requête ou dans une base de données de graphes. Ainsi, plusieurs approches ont été proposées pour répondre aux requêtes de recherche de graphes par similarité (ou simplement, requêtes par similarité). Voir [71, 119, 133, 146].

Dans les deux catégories citées ci-dessus, l'appariement de graphes ou de sous-graphes est une opération fondamentale dans le processus de recherche de graphes. Cet appariement est réalisé en effectuant un ensemble de tests d'isomorphisme entre le graphe requête et le(s) graphe(s) de la base de données (voir la section 1.3 page 13). Cependant, la recherche d'un isomorphisme de graphe ou de sous-graphe est NP-difficile [56]. Ainsi, pour réduire l'espace de recherche et rendre plus acceptable la complexité temporelle du processus d'évaluation d'une requête de graphe, la plupart des algorithmes proposés utilisent des techniques d'indexation des caractéristiques de la base de données de graphes. En outre, une recherche sur la base toute entière est écartée.

Plusieurs méthodes d'indexation et d'évaluation de requêtes de graphes ont été développées pour résoudre les deux problèmes de recherche de graphes cités ci-dessus. La figure 1.9 fournit une description synthétique de ces méthodes ainsi que le type d'appariement utilisé.

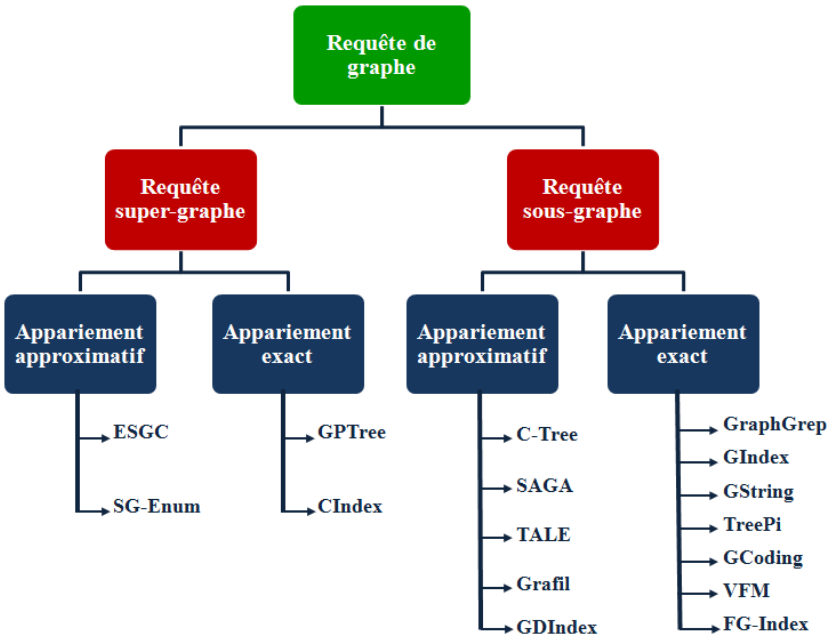



FIGURE 1.9 – Exemples d'approches d'évaluation de requêtes de graphes.

Dans ce qui suit, deux approches d'évaluation dédiées, respectivement, aux requêtes sous-graphes et aux requêtes super-graphes, sont présentées. En particulier, pour chacune de ces approches, l'unité d'indexation utilisée et le principe d'évaluation des requêtes de graphes sont abordés.

### 1.5.1 Requêtes sous-graphes

Plusieurs méthodes d'indexation et d'évaluation de requête ont été proposées pour résoudre les deux problèmes de recherche de graphes cités auparavant (voir la figure 1.9). Bien que la recherche sur l'appariement de graphes possède un très grand historique, la plupart de ces travaux [77, 94, 120, 133, 145, 29, 140, 132, 71, 146] ont porté sur l'appariement de sous-graphe. Comme nous pouvons le constater sur la figure 1.9, de nombreux efforts de recherche se sont focalisés sur l'évaluation de requêtes sous-graphes dans le but d'améliorer non seulement l'espace mémoire utilisé mais aussi le temps de réponse. Les approches proposées peuvent être scindées en deux classes (voir la figure 1.9) :

- Méthodes fondées sur l'appariement exact (i.e. la recherche exacte de graphes contenant complètement le graphe requête).
- Méthodes fondées sur l'appariement approximatif (i.e., la recherche par similarité des graphes contenant approximativement le graphe requête).

Les approches de ces deux classes seront abordées dans ce qui suit.

#### 1.5.1.1 Approches fondées sur l'appariement exact

Plusieurs approches [77, 120, 145, 29, 140, 146, 94, 28] ont été proposées pour supporter les requêtes sous-graphes. Dans cette sous-section, nous présentons le modèle *GraphGrep* comme exemple de telles approches et résumons les travaux les plus notoires sur le sujet.

– **L'approche GraphGrep**

Giugno et Shasha ont développé une méthode d'indexation et d'évaluation de requêtes sous-graphes basée sur la notion de *chemins*. Cette approche, appelée *GraphGrep*, permet d'indexer les graphes d'une base de données de graphes en énumérant tous leurs chemins de longueur inférieure ou égale à une certaine taille fixée par l'utilisateur (notée $maxL$). Ensuite, elle utilise l'index pour identifier tous les graphes de la base contenant tous les chemins (inférieur ou égale à $maxL$) de la requête. Cependant, la taille de l'index n'est pas exponentielle avec la taille des graphes mais exponentielle avec la taille des chemins.

**(i) Évaluation de requêtes**

Afin d'évaluer une requête sous-graphe, *GraphGrep* applique les trois phases d'interrogation de bases de données citées dans la section 1.4 page 24. En phase de construction d'index, *GraphGrep* énumère, pour chaque graphe de la base de données, tous les chemins existants d'une taille inférieure ou égale à la valeur du paramètre $maxL$ fixé par l'utilisateur. Puis, il enregistre le nombre d'occurrences de chaque chemin du graphe dans une table de hachage ayant comme clé les étiquettes des chemins constitués.

Par ailleurs, *GraphGrep* procède à une analyse de la requête considérée afin de construire sa table de hachage et de générer l'ensemble des patterns constituant la requête. Ces patterns décrivent d'une manière compacte tous les chemins de la requête et permettent une reconstitution de son graphe.

La base de données est ensuite filtrée en comparant sa table de hachage à celle de la requête. Tout graphe contenant au moins une valeur dans sa table inférieure à celle de la table du graphe de la requête, est éliminé.

Enfin, *GraphGrep* effectue un appariement de sous-graphes entre les graphes trouvés dans la phase de filtrage et le graphe de la requête.

#### (ii) Analyse de l'approche

En analysant l'approche *GraphGrep* présentée dans [60], nous constatons qu'elle a deux avantages principaux : (i) l'utilisation de la notion de chemin qui est facile à manipuler et ; (ii) la construction d'un index de taille prédéfinie et linéaire avec la taille de la base de données de graphes.

Cependant, *GraphGrep* souffre de quelques inconvénients qui ne lui permettent pas d'être utilisé dans le cas des requêtes de graphes plus complexes. Le premier inconvénient consiste en son utilisation des chemins qui sont une structure tellement simple qu'ils ne permettent pas de sauvegarder des informations sur la structure des graphes. Enfin, le deuxième inconvénient réside dans le fait que le nombre des chemins calculés pour chaque graphe augmente exponentiellement avec la taille $maxL$.

### – Bilan et discussion

De nombreux travaux ont été réalisés pour répondre aux requêtes sous-graphes. Parmi les approches proposées, nous avons étudié et résumé leurs propriétés les plus importantes dans le tableau 1.3.

TABLE 1.3 – Tableau récapitulatif des approches d'appariement exact de requêtes sous-graphes.

| **Méthodes** | **Unité d'indexation** | **Points forts** | **Points faibles** |
|---|---|---|---|
| GraphGrep | Chemins | – Construction et manipulation facile de l'index.<br>– Taille de l'index linéaire avec la taille de la base. | – Taille de l'index exponentielle avec maxL.<br>– Information structurelle non préservée.<br>– Inadaptée aux graphes de grande taille. |
| GIndex | Fragments fréquents | – Index plus compact.<br>– Réduction des fragments fréquents à indexer.<br>– Grande puissance d'élagage. | – Processus de construction d'index très coûteux.<br>– Ensemble $C_q$ de grande taille.<br>– Beaucoup de fausses réponses.<br>– Phase de vérification coûteuse en termes de temps. |
| GString | Séquences | – Comparaison lexicographique au lieu de tests d'isomorphisme. | – Une approche ad hoc.<br>– Mieux adaptée aux graphes avec un petit nombre de structures de base. |
| VFM | Nœuds importants | – Taille de l'index réduite et linéaire avec celle de la base.<br>– Pas d'énumération des fragments de la requête. | – Indexation dépendante de l'importance des nœuds.<br>– Possibilité d'élimination de réponses positives. |

Toutes les approches [29, 60, 77, 94, 120, 145] consignées dans le tableau 1.3 répondent à la problématique de recherche exacte d'un sous-graphe (requête) dans une base de données de graphes. Cependant, il n'existe aucune garantie de fournir une réponse précise à cette recherche.

Pour cela, nous nous intéressons, dans ce qui suit, à la recherche approximative de graphes (voir la figure 1.9) pour pallier les manques des méthodes de recherche exacte (tels que les réponses vides ou en nombre insuffisant) ou encore s'adapter aux applications qui nécessitent une recherche par similarité telles que l'imagerie médicale, la bio-informatique, la chimie, etc.

#### 1.5.1.2 Approches fondées sur l'appariement approximatif

Plusieurs domaines d'application nécessitent une recherche de graphes fondée sur le paradigme d'appariement approximatif, pour les trois raisons suivantes : (i) souvent les graphes de ces applications sont de nature incomplète et bruitée comme la bio-informatique ; (ii) des applications récentes préfèrent des résultats basés sur l'appariement approximatif plutôt que ceux basés sur l'appariement exact car ces résultats peuvent apporter plus d'information sur ce qui pourrait manquer dans la requête ou dans la base et aussi ; (iii) pour traiter les requêtes à réponses vides suite à une recherche basée sur l'appariement exact.

Dans la littérature, plusieurs approches [132, 133, 71, 140, 146] ont été développées pour supporter les requêtes de recherche par similarité. Parmi ces méthodes, nous présentons le modèle *Tale* [133].

– **L'approche Tale**

*Tale*, proposé par Tian et Patel [133], est une méthode d'évaluation approximative de requêtes de graphes de grande taille. La méthode d'indexation utilisée, appelée *NH-Index*, est basée sur la notion de *voisinage* des nœuds les plus importants des graphes de la base de données. Cette notion permet à *Tale* d'incorporer les informations structurelles des graphes dans l'index et ainsi d'augmenter la puissance d'élagage.

Contrairement aux méthodes d'indexation existantes où la taille de l'index croît exponentiellement avec la taille des fragments considérés, *Tale* utilise la notion de voisinage des nœuds importants lui permettant non seulement de capturer la structure locale autour de chaque nœud mais aussi d'avoir un index dont la taille est linéaire avec celle de la base de données.

Le voisinage est défini comme étant un *sous-graphe induit* d'un nœud et ses voisins (i.e., les nœuds adjacents). Il utilise un appariement exact et, en cas d'échec, il fait appel à une variante d'appariement qui utilise une technique d'approximation. Pour cela, *Tale* introduit une certaine flexibilité sur l'appariement des nœuds les moins importants par application d'un appariement approximatif.

En effet, l'utilisateur introduit un degré d'approximation « $\rho$ » qui définit le pourcentage des voisins d'un nœud de la requête qui peuvent être absents ou avoir de fausses correspondances avec les voisins d'un nœud de la base de données. Pour mesurer la qualité de cet appariement, *Tale* calcule un degré de qualité en tenant compte du nombre de voisins manquants et du nombre de connexions manquantes entre voisins.

**(i) Évaluation de requêtes**

Afin d'évaluer une requête sous-graphe, *Tale* sélectionne d'abord les nœuds les plus importants de la requête, i.e., les nœuds ayant un nombre de voisins supérieur ou égal à un seuil minimal de voisins, noté $nb_{min}$. Puis, il filtre les graphes de la base de données qui contiennent les mêmes nœuds importants que ceux de la requête, en exploitant l'index NH-Index.

Ensuite, il étend son filtrage en effectuant l'appariement exact ou approximatif (selon le choix de l'utilisateur) des nœuds restants de la requête avec ceux des graphes candidats résultants de l'étape

précédente.

Enfin, en cas d'appariement approximatif, *Tale* retourne les K meilleures réponses en classant les graphes sélectionnés par ordre décroissant suivant leurs degrés de similarité avec le graphe de la requête.

#### (ii) Analyse de l'approche

Le modèle *Tale* se distingue des autres approches par le fait qu'il effectue un appariement guidé par l'importance des nœuds, ce qui renforce son pouvoir d'élagage. Il supporte les deux recherches exacte et approximative de requêtes sous-graphes. De plus, il offre une grande flexibilité lors des calculs de similarité en permettant le choix des paramètres en entrée et le choix de la méthode de calcul de similarité à utiliser. Pour toutes ces raisons, *Tale* constitue une approche pouvant s'adapter aisément aux exigences des différentes applications.

L'inconvénient de *Tale* réside dans le choix de la valeur du nombre minimal ($nb_{min}$) de voisins qu'un nœud important doit avoir. En effet, le choix d'une valeur quelconque peut engendrer une perte d'information sur certains graphes de la base de données, car les graphes possédant des nœuds de valeur inférieure à $nb_{min}$ ne seront pas indexés.

– **Bilan et discussion**

Dans le tableau 1.5, nous résumons les propriétés principales des approches d'évaluation de requêtes sous-graphes appliquant un appariement approximatif au lieu d'un appariement exact.

### 1.5.2 Requêtes super-graphes

Comme cité dans la section 1.5 page 25, une autre famille d'approches d'évaluation de requêtes de graphes a été proposée pour répondre aux besoins des applications nécessitant une recherche de requêtes super-graphes plutôt que sous-graphes. En effet, ce type de requêtes est très important dans plusieurs domaines d'application. En chimie, par exemple, des composantes chimiques, dites descripteurs, sont stockées avec leurs propriétés chimiques dans une base de données de graphes. À la découverte d'une nouvelle molécule, les chimistes cherchent souvent à trouver les descripteurs contenus dans la nouvelle molécule pour prédire les propriétés possibles de cette dernière. Pour cela, les chimistes peuvent émettre une requête super-graphe représentant la nouvelle molécule sur la base de données de descripteurs.

Contrairement aux requêtes sous-graphes, peu de travaux [26, 119, 134, 157] ont été réalisés pour répondre aux requêtes super-graphes. Toutefois, les approches proposées peuvent aussi être divisées en deux classes (voir la figure 1.9 page 27) :

– Approches fondées sur l'appariement exact (i.e., la recherche exacte de graphes complètement contenus dans le graphe requête).
– Approches fondées sur l'appariement approximatif (i.e., la recherche par similarité des graphes approximativement contenus dans le graphe requête).

TABLE 1.5 – Tableau récapitulatif des approches d'appariement approximatif de requêtes sous-graphes.

| **Méthode** | **Unité d'indexation** | **Points forts** | **Points faibles** |
|---|---|---|---|
| Tale | Nœuds importants | – Grande puissance d'élagage.<br>– Index de taille linéaire avec celle de la base de données.<br>– Approche performante pour les requêtes de grande taille.<br>– Pas de phase de vérification. | – Impact du choix du nombre de voisins minimal d'un nœud important « $nb_{min}$ » sur la perte d'informations sur certains graphes. |
| Grafil | Fragments | – Grande puissance d'élagage.<br>– Précision et efficacité. | – Processus d'extraction des fragments très coûteux. |
| SAGA | Fragments | – Grande puissance d'élagage.<br>– Efficacité sur les graphes de petite taille.<br>– Flexibilité sur les composants du modèle de calcul de distance. | – Approche très coûteuse pour les graphes de grande taille.<br>– Méthode restreinte au domaine de la biologie. |
| GDIndex | Arbre | – Pas de procédure d'énumération de fragments.<br>– Pas de phase de vérification.<br>– Construction et manipulation facile de l'arbre. | – Approche inadéquate pour les graphes de grande taille. |
| C-Tree | Arbre | – Pas de procédure d'énumération de fragments.<br>– Structure d'index similaire à R-tree.<br>– Pseudo isomorphisme de sous-graphe.<br>– Flexibilité dans la méthode de calcul de similarité. | – Construction et mise à jour de C-tree très coûteuse.<br>– Construction de C-tree dépendant de l'ordre d'insertion des graphes. |

Dans ce qui suit, nous présentons d'abord les approches de la première classe. Les approches de la seconde classe, seront introduites dans la sous-section 1.5.2.2 page 36.

#### 1.5.2.1 Approches fondées sur l'appariement exact

Afin de répondre aux requêtes super-graphes, des approches basées sur l'appariement exact de sous-graphes, ont été proposées [26, 157]. Nous résumons, dans ce qui suit, les principales propriétés de ces approches et détaillons une d'entre elles, à savoir *CIndex* [26].

– **CIndex**

Chen *et al.* [26] ont développé un algorithme, dit *CIndex*, permettant de rechercher les graphes d'une base de données contenus dans un graphe requête. *CIndex* est une technique d'indexation compacte basée sur les notions de sous-graphes (fragments) et de logique d'exclusion [6]. L'idée de base de cette approche est de capturer la différence entre les graphes de la base de données et les graphes requêtes.

**(i) Évaluation de requêtes**

L'évaluation de requêtes super-graphes présentée dans [26] est réalisée en appliquant les trois phases citées dans la section 1.4 page 24. En se basant sur l'historique des requêtes de graphes, *CIndex*, construit l'index composé des caractéristiques (i.e., fragments, sous-graphes) de la base de données apparaissant rarement dans les graphes requêtes. Ensuite, il effectue des tests d'isomorphisme de sous-graphes entre les fragments de l'index et le graphe requête, pour retourner l'ensemble des fragments qui ne sont pas contenus dans la requête. Cet ensemble permet donc d'éliminer les graphes contenant au moins un de ces fragments pour calculer par la suite les graphes de la base de données candidats à la requête. Enfin, *CIndex* vérifie l'inclusion de chaque graphe candidat dans le graphe de la requête en effectuant des tests d'isomorphisme de sous-graphe.

**(ii) Analyse de l'approche**

En utilisant la méthode de filtrage-vérification, le processus d'évaluation de requêtes super-graphes de *CIndex* permet d'éviter un grand nombre de tests d'isomorphisme de sous-graphes. En outre, la taille de l'index construit est très petit, car les caractéristiques le composant sont filtrées en utilisant l'historique des requêtes de graphes.

Néanmoins, la performance et l'efficacité de *CIndex* dépend de l'historique des requêtes super-graphes déjà traitées. Cet historique peut fréquemment changer au fil du temps et l'index des caractéristiques peut donc être assez souvent obsolète. Toutefois, la performance de *CIndex* se dégrade considérablement avec le temps, car le suivi et la mise à jour de son index nécessitent un grand

6. La logique d'exclusion signifie que si un fragment (sous-graphe) $f$ de la base de données $D$ n'est pas contenu dans le graphe requête $q$, i.e. $f \nsubseteq q$, et $f$ est un sous-graphe d'un graphe $g$ de la base $D$, i.e. $f \subseteq g$, alors $g$ est écarté de la recherche car $q \not\supseteq g$.

nombre de tests d'isomorphisme de sous-graphe. De plus, *CIndex* ne permet pas de traiter les réponses vides dans le cas où aucun graphe de la base de données n'est complètement contenu dans le graphe requête.

– **Bilan et discussion**

Les approches [26, 157] présentées dans le tableau 1.7, permettent donc de traiter le problème de recherche exacte de requêtes super-graphes. Néanmoins, ces modèles ne répondent pas forcément aux besoins des utilisateurs et de certaines applications, comme éviter les réponses vides ou bien retrouver des objets similaires à un autre objet. À titre d'exemple, dans le domaine de vidéo, l'utilisateur peut avoir besoin de retrouver des images similaires à une image donnée pour lui permettre par la suite de reconstruire une scène donnée.

Dans la littérature, peu de méthodes basées sur la recherche approximative de graphes pour répondre aux requêtes super-graphes (voir la figure 1.9), ont été présentées. Dans la sous-section suivante, nous introduisons brièvement ces approches permettant non seulement de pallier les manques des méthodes de recherche exacte mais aussi de s'adapter aux applications qui nécessitent une recherche par similarité de graphes comme dans le domaine de la chimie, de l'imagerie, de services Web, ...

TABLE 1.7 – Tableau récapitulatif des approches d'appariement exact de requêtes super-graphes.

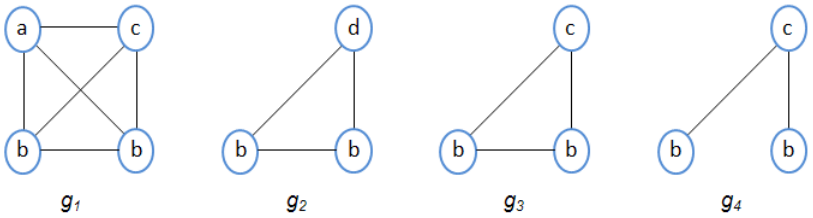
| **Méthode** | **Unité d'indexation** | **Points forts** | **Points faibles** |
|---|---|---|---|
| CIndex | Fragments | – Index compact.<br>– Grande puissance d'élagage.<br>– Réduction du nombre de tests d'isomorphisme de sous-graphe. | – Processus d'indexation des fragments dépendant de l'historique des requêtes super-graphes.<br>– Performance et efficacité de l'approche dépendantes de la fiabilité de l'index.<br>– Suivi et mise à jour de l'index très coûteux. |
| GPTree | Fragments | – Organisation compacte des graphes de la base de données.<br>– Comparaison simultanée entre plusieurs graphes candidats et le graphe requête. | – Processus d'indexation nécessitant plusieurs tests d'isomorphisme de sous-graphe.<br>– Suivi et mise à jour de l'index très coûteux. |

#### 1.5.2.2 Approches fondées sur l'appariement approximatif

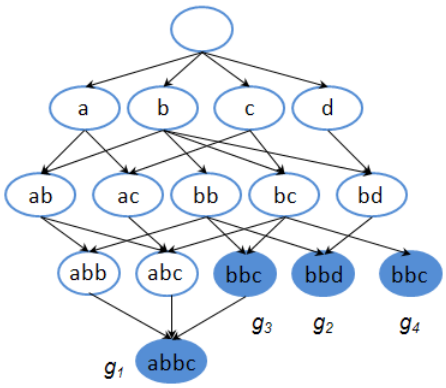
Peu de travaux ont été proposés pour répondre aux requêtes super-graphes, au moyen d'un appariement approximatif. Parmi ces approches, nous citons les modèles *ESGC* [134] et *SG-Enum* [119].

– **ESGC**

Dans [134], les auteurs proposent un algorithme, dit *ESGC*, basée sur la décomposition de graphes (DAG) *en sous-graphes induits* et connectés, pour répondre aux problèmes de recherche exacte et approximative de requêtes super-graphes. Pour cela, un index est construit en deux étapes.
La première consiste en l'exécution du processus de décomposition des graphes de la base de données permettant de créer un graphe acyclique non-orienté, noté DAG (i.e., Direct Acyclic Graph), qui contient un nœud pour chaque sous-graphe induit unique des graphes de la base de données (voir la figure 1.10 extraite de [140]). La deuxième étape construit la table de hachage de cette décomposition permettant ainsi l'indexation des sous-graphes énumérés durant la décomposition des graphes. Les clés de la table de hachage construite sont calculées en utilisant les codes basés sur la forme canonique des matrices d'adjacences des graphes. Ainsi, en utilisant cette méthode, tous les graphes isomorphes produisent la même clé de hachage.
À l'aide de cette décomposition, le nombre de tests d'isomorphisme de sous-graphe est donc minimisé en réduisant la taille de l'ensemble des graphes candidats à la requête super-graphe.




(a) Une base de données de graphes $D$.




(b) La décomposition DAG de la base $D$.

FIGURE 1.10 – Exemple de décomposition de graphes DAG.

**(i) Évaluation de requêtes**

Pour répondre à une requête super-graphe, Tong et Zhong procèdent comme suit. L'index de l'algorithme *ESGC* est d'abord construit pour générer l'arbre DAG (i.e., Direct Acyclic Graph)de la base de données de graphes ainsi que la table de hachage qui lui correspond. De la même manière, l'arbre DAG et la table de hachage de la requête sont ainsi construits.

Ensuite, l'algorithme *ESGC* vérifie s'il existe des clés de la table de hachage de la base de données qui sont des entrées dans celle de la requête. Cette vérification entre les clés de hachage permet à *ESGC* de filtrer l'ensemble des graphes candidats au graphe de la requête.

Enfin, des tests d'isomorphisme de sous-graphe sont effectués entre les graphes candidats de la base de données et le graphe requête. Ainsi, les graphes complètement inclus dans la requête sont déterminés. Dans le cas de réponse vide, l'algorithme *ESGC* fournit les graphes approximativement inclus dans le graphe requête.

**(ii) Analyse de l'approche**

Le modèle *ESGC* permet donc de déterminer l'ensemble des graphes complètement ou approximativement inclus dans le graphe d'une requête donnée. En utilisant la notion de décomposition de graphes, les sous-graphes communs entre les graphes de la base de données sont sauvegardés une seule fois dans l'index. Par conséquent, l'algorithme *ESGC* possède une meilleure performance que CIndex [26] dans le cas d'une recherche exacte de requêtes super-graphes.

Cependant, ESGC devient moins efficace avec l'augmentation du paramètre, noté $\delta$, permettant de fixer le nombre maximal de nœuds manquants dans chaque entrée de la table de hachage de la requête. De plus, l'application de la décomposition de graphes est limitée aux graphes de petite taille.

– **Bilan et discussion**

Dans le tableau 1.9, nous présentons un récapitulatif, des approches ESGC et SG-Enum pour évaluer les requêtes super-graphes, ainsi que leurs principales caractéristiques.

TABLE 1.9 – Tableau récapitulatif des approches d'appariement approximatif de requêtes super-graphes.

| **Méthode** | **Unité d'indexation** | **Points forts** | **Points faibles** |
|---|---|---|---|
| ESGC | Arbre | – Index de taille réduite.<br>– Manipulation facile de l'arbre.<br>– Pas de processus d'énumération de fragments.<br>– Réduction du nombre de tests d'isomorphisme de sous-graphe. | – Détérioration de la performance et de l'efficacité de l'approche avec l'augmentation de la valeur du paramètre $\delta$.<br>– Approche inadéquate pour les graphes de grande taille. |
| SG-Enum | Arbre | – Organisation compacte des graphes de la base de données.<br>– Manipulation facile de l'arbre.<br>– Pas de processus d'énumération de fragments.<br>– Pas de processus de détection de sous-graphe commun maximal. | – Construction d'index très coûteuse.<br>– Index dépendant du paramètre $\sigma$.<br>– Pas de phase de filtrage.<br>– Possibilité de la non-correspondance de la valeur de $\sigma$ aux besoins de l'utilisateur.<br>– Approche très coûteuse pour des graphes et bases de données de grande taille. |

## 1.6 Conclusion

L'objectif principal du présent chapitre a été l'introduction des principaux concepts liés à notre travail de recherche, à savoir les graphes et les techniques d'interrogation de bases de données de graphes.

L'interrogation d'une base de données de graphes nécessite un processus d'appariement, pour comparer le graphe de la requête de l'utilisateur avec les graphes de la base de données interrogée. Deux types d'appariement de graphes ont été introduits dans ce chapitre : (i) un appariement exact et (ii) un appariement approximatif. Le premier exige soit une égalité entre les deux graphes comparés, soit une inclusion totale du plus petit graphe dans le plus grand. L'appariement approximatif quant à lui, offre une tolérance par rapport au nombre d'erreurs de mise en correspondance entre les nœuds et/ou les arêtes des deux graphes appariés.

Dans cette dernière approche, il est toujours intéressant d'évaluer la distance entre graphes,

pour ordonner les réponses par similarité décroissante au graphe de la requête. Pour cela, nous avons présenté les mesures de calcul de distance entre graphes les plus utilisées : (i) la distance d'édition basée sur les opérations d'édition et une fonction de coût ; (ii) la distance basée sur le sous-graphe commun maximal ; (iii) la distance basée sur l'union de graphes et (iv) la distance basée sur le super-graphe commun minimal. Chaque méthode a été discutée, l'accent a été mis sur les caractéristiques qui peuvent influencer positivement ou négativement le processus d'évaluation de requêtes de graphes.

Nous avons également décrit les différentes étapes et techniques usuellement utilisées dans le processus d'interrogation de bases de données de graphes, pour réduire l'espace de recherche et assurer une bonne performance en termes de temps et d'espace mémoire. Une de ces étapes consiste en l'indexation des graphes de la base. Il nous est alors apparu opportun de présenter quelques exemples d'unités d'indexation à titre illustratif, tels : les séquences, les chemins, les fragments fréquents, ...

Des approches d'interrogation de bases de données de graphes permettant d'évaluer les requêtes utilisateurs ont été aussi étudiées dans ce chapitre. On distingue alors : (i) la famille des requêtes sous-graphes et (ii) celle des requêtes super-graphes. La première recherche tous les graphes de la base de données qui contiennent le graphe de la requête. Quant à la deuxième, elle vise à retrouver tous les graphes de la base qui sont contenus dans le graphe requête. Chacune des deux familles se divisent en deux catégories en appliquant soit l'appariement exact, soit l'appariement approximatif pour comparer deux graphes. Cette étude met en évidence que toutes les approches existantes ont eu recours à des techniques permettant d'améliorer les performances de l'appariement. Plus précisément, ces approches ont proposé des stratégies qui permettent d'améliorer le temps de traitement en réduisant, par exemple, l'espace de recherche. Nous remarquons également, et pour autant que nous le sachions, qu'il n'existe aucun travail ayant étudié la pertinence des résultats vis-à-vis de l'utilisateur. De plus, la prise en considération des contraintes et des préférences des utilisateurs n'a été que très peu abordée dans la littérature.

En effet, les méthodes présentées dans ce chapitre s'appuient uniquement sur les caractéristiques structurelles communes entre les graphes de la base de données interrogée et le graphe requête. Cependant, ces approches deviennent moins performantes et moins pertinentes pour une base de données composée d'un grand nombre de graphes structurellement similaires au graphe de la requête (car plusieurs graphes candidats sont sélectionnés et ainsi de nombreuses mises en correspondance vont être effectuées entre les graphes candidats et le graphe requête, ce qui rend la phase de filtrage et celle de vérification moins efficaces et moins pertinentes.).

Pour pallier cet inconvénient, des préférences utilisateur peuvent être intégrées dans le processus d'interrogation de bases de données de graphes, pour une amélioration de la phase de filtrage en éliminant non seulement les graphes structurellement différents au graphe de la requête mais aussi ceux qui ne satisfont pas les contraintes de l'utilisateur. De plus, l'évaluation de ces préférences permet d'ordonner les graphes du point de vue de l'utilisateur.

Avant de présenter nos contributions dans ce contexte et qui feront l'objet de la deuxième partie

de ce manuscrit, nous présentons dans le chapitre suivant, un état de l'art sur les méthodes d'interrogation *à préférences* des bases de données relationnelles, ce qui permet d'apporter une plus grande *flexibilité* dans les requêtes des utilisateurs.

# Chapitre 2

# Sur les requêtes à préférences

## 2.1 Introduction

D'une manière générale, une préférence est une expression permettant de définir un ordre sur les éléments considérés comme les plus pertinents. Dans le contexte d'interrogation de bases de données, ces préférences permettent d'augmenter les requêtes des utilisateurs pour sélectionner et ordonner les éléments souhaités des plus satisfaisants aux moins satisfaisants, ce qui est particulièrement utile lorsqu'un grand nombre de réponses est retourné. Ce type de requêtes, appelés *requêtes à préférences* ou encore *requêtes flexible*s, permet donc une interrogation *flexible* des bases de données en évitant les réponses vides et pléthoriques et en triant les réponses selon leurs pertinences par rapport aux préférences de l'utilisateur.

En effet, dans le cadre des bases de données, les préférences permettent d'améliorer la capacité d'expression des langages de requêtes et faciliter l'accès aux informations pertinentes [16, 80]. Dans cet objectif, de nombreux travaux ont été réalisés [14, 30, 39, 40, 45, 47, 48, 70, 150, 151, 152] dont une partie de ces travaux considère des préférences exprimées à l'aide de prédicats flous, dits aussi prédicats graduels [15], tels que *pas cher*, *rapide*, *proche*, ...

Une requête flexible consiste donc en une ou plusieurs conditions floues de sélection (i.e., préférences floues) sur les valeurs des attributs. Ces préférences sont généralement représentées sous forme d'une conjonction et/ou disjonction de conditions. Chacune d'entre elles peut être associée à un degré représentant son niveau d'importance par rapport aux autres préférences du point de vue de l'utilisateur.

L'objectif de ce chapitre est de donner un aperçu sur les modèles de préférences dans le cadre des bases de données. Les différentes approches étudiées sont classées en deux familles selon que l'approche est de nature agrégative ou non. La famille des modèles agrégatifs est fondée sur la théorie des ensembles flous et sur la propriété de commensurabilité [1] des degrés de satisfaction d'un élément par rapport aux préférences d'une requête utilisateur. Les degrés de satisfaction aux préférences d'un élément peuvent être agrégés en un seul degré global et ainsi, tous les éléments peuvent être

1. La propriété de commensurabilité est vérifiée si et seulement si les degrés de satisfaction résultant de différentes préférences sont comparables et basés sur la même échelle.

ordonnés du plus satisfaisant au moins satisfaisant. Quant aux approches non-agrégatives, les degrés de satisfaction des préférences de la requête utilisateur sont regroupés dans des vecteurs utilisés par la suite pour sélectionner les éléments non-dominés au sens de Pareto.

Avant de présenter ces modèles, nous définissons, en section 2.2, les prédicats à préférences ainsi que les deux types de préférences, à savoir, les préférences unipolaires et les préférences bipolaires. Ensuite, les modèles agrégatifs basés sur les ensembles flous sont introduits en section 2.3 et, ceux basés sur l'opérateur de Pareto sont résumés dans la section 2.4. Enfin, nous terminons ce chapitre par une conclusion, dans laquelle une synthèse sur les différentes notions étudiées est présentée.

## 2.2 Prédicats à préférence

Les prédicats à préférence dits flous ou graduels, sont utilisés pour exprimer des requêtes avec des préférences graduelles plutôt que booléennes, où les conditions ne sont plus « vraies ou fausses » mais « plus ou moins satisfaites ». Par exemple, dans la requête « retrouver un hôtel *pas cher* et aussi *proche* que possible de la plage », décrit une préférence composée de deux conditions élémentaires (dites aussi préférences atomiques) sur les attributs ou critères : prix et distance, en utilisant respectivement, les prédicats graduels « *pas cher* » et « *proche* ».

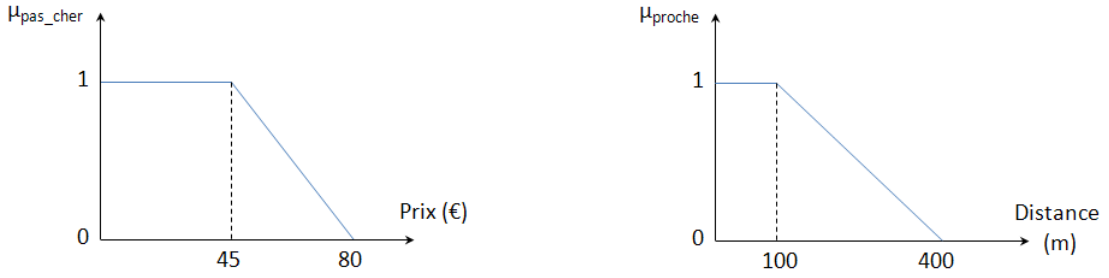
Pour répondre à de telles requêtes, les prédicats flous composant ses conditions élémentaires peuvent être décrits au moyen des ensembles flous [45] (voir l'annexe A). Cette modélisation permet ainsi de déterminer le degré de satisfaction d'un élément, par rapport à un prédicat donné. Il s'agit de fonctions d'appartenance qui ont la capacité de s'adapter au contexte et à la subjectivité [2] de l'utilisateur. Ces fonctions sont de $d_i$ dans $[0, 1]$, permettant de décrire les préférences de l'utilisateur sur chaque domaine d'attribut, $d_i$, présent dans la requête. Le degré d'appartenance est interprété comme le niveau de satisfaction ou de préférence. En pratique, les fonctions d'appartenance sont généralement représentées par des trapèzes (éventuellement ouverts à gauche ou à droite) ou des triangles [15] (voir la figure 2.1). L'avantage principal de ces fonctions est leur capacité d'adaptation au contexte et à la subjectivité de l'utilisateur.

Dans l'exemple précédent, les prédicats « *pas cher* » et « *proche* » peuvent être décrits par des ensembles flous, permettant d'obtenir pour un prix et une distance donnés, des degrés de satisfaction définis sur l'intervalle $[0, 1]$. Ces degrés sont commensurables et peuvent donc être combinés et agrégés.

La figure 2.1 illustre deux exemples de fonctions d'appartenance des deux prédicats « pas cher » et « proche » de l'exemple précédent. La fonction d'appartenance de « pas cher » est représentée par un trapèze ouvert à gauche, décrit par le quadruplet $(\alpha, \beta, \varphi, \psi)$ où $\alpha = 0$€, $\beta = 0$€, $\varphi = 45$€et $\psi = 80$€. Pour toute valeur $x$ appartenant à l'intervalle $[0, 45]$, son degré d'appartenance (noté $\mu_{pas\_cher}$) au prix pas cher est égal à 1, et plus le prix $x$ est supérieur à 45€, plus son degré d'appartenance au prix pas cher diminue.

2. La propriété de subjectivité désigne le caractère de ce qui varie selon la personnalité, l'opinion, les principes, les goûts et les objectifs de chacun, en opposition à « objectif » qui tient de la neutralité.

De la même manière, la fonction d'appartenance de « *proche* » est modélisée par un trapèze ouvert à gauche, décrit par le quadruplet $(\alpha, \beta, \varphi, \psi)$ où $\alpha = 0$m, $\beta = 0$m, $\varphi = 100$m et $\psi = 400$m. Ainsi, l'utilisateur est complètement satisfait lorsque la distance $x$ entre l'hôtel et la plage ne dépasse pas 100m, mais plus la distance $x$ est supérieure à 100m, moins l'utilisateur est satisfait (i.e. le degré d'appartenance (noté $\mu_{proche}$) de $x$ à une distance proche de la plage diminue et se rapproche de 0).




(a) La fonction d'appartenance du prédicat « pas cher ».

(b) La fonction d'appartenance du prédicat « proche ».

FIGURE 2.1 – Exemples de fonctions d'appartenance.

Il est aussi important de préciser qu'il est possible d'appliquer des modificateurs sur la fonction d'appartenance d'un ensemble flou pour décrire un adverbe du langage naturel, comme *assez*, *très*, *plus ou moins*, ... Ces modificateurs sont modélisés par des fonctions de $[0, 1]$ dans $[0, 1]$. Par exemple, le prédicat flou « *très proche* » peut être calculé à partir de la fonction d'appartenance du prédicat « *proche* » en lui appliquant un concentrateur.

Les degrés de satisfaction individuels associés à ces conditions peuvent être combinées en utilisant une panoplie de connecteurs flous. Ces connecteurs peuvent être des agrégations conjonctives ou disjonctives (éventuellement pondérées pour exprimer des niveaux d'importance entre critères), des moyennes (exprimant un compromis entre critères) ou des quantificateurs flous (tels que la plupart, environ, la moitié, ...) lorsque seules quelques conditions élémentaires d'une requête peuvent être satisfaites. Ainsi, les résultats de la requête sont évalués en fonction de leur adéquation aux critères de sélection et peuvent être ordonnés des plus satisfaisants aux moins satisfaisants du point de vue de l'utilisateur.

Dans ce qui suit, nous définissons deux types de préférences floues selon que les conditions exprimées dans la requête sont unipolaires ou bipolaires.

### 2.2.1 Préférence unipolaire

Une préférence unipolaire est une expression composée d'un seul type de conditions élémentaires, vérifiant la propriété de commensurabilité. Pour chaque élément $x$, ses degrés de satisfaction sont évalués en utilisant les fonctions d'appartenance correspondantes à ces conditions [15, 45]. Ensuite, ces degrés sont combinés et agrégés à l'aide des connecteurs flous pour déterminer le degré global de satisfaction de chaque élément $x$ par rapport aux conditions de la préférence unipolaire de l'utilisateur [45, 125, 142].

Plusieurs connecteurs ont été proposés dans la théorie des ensembles flous. Ces connecteurs peuvent être des agrégations conjonctives ou disjonctives (éventuellement pondérées pour exprimer des niveaux d'importance entre critères), des moyennes (exprimant un compromis entre critères) ou des quantificateurs flous (tels que la plupart, environ, la moitié, ...) lorsque seules quelques conditions élémentaires d'une requête doivent être satisfaites.

Enfin, un pré-ordre total peut être appliqué sur l'ensemble des éléments suivant leur degré global exprimant leur adéquation aux conditions élémentaires de la préférence unipolaire de l'utilisateur. Voir [45, 15], pour plus de détails.

### 2.2.2 Préférence bipolaire

Les conditions bipolaires représentent un autre type de préférences, permettant d'assurer une interrogation flexible des bases de données. De telles préférences permettent de distinguer entre *les conditions positives* et *les conditions négatives*. Les conditions négatives expriment, ce qui est plus ou moins indésirable, voire complètement rejeté. Les conditions positives expriment les valeurs satisfaisantes et souhaitées par l'utilisateur. Par exemple, la requête « *rechercher les hôtels de préférences proches de la plage et rejeter ceux qui sont chers* », exprime bien une condition de nature bipolaire dans laquelle on distingue ce qui est rejeté (i.e. *les hôtels chers*), de ce qui est souhaité (i.e. *les hôtels proches de la plage et pas chers*) par l'utilisateur.

De ce fait, nous déduisons que les requêtes bipolaires, associant ces deux types de conditions (négative et positive), ne permettent pas seulement d'exprimer les exigences de l'utilisateur d'une manière plus intuitive, mais aussi de réduire le taux de sélectivité en éliminant les éléments qui ne satisfont pas ses conditions.

Dans le contexte des requêtes flexibles, plusieurs travaux [13, 152] expriment une préférence comme étant une combinaison de conditions négatives et positives. La condition négative, appelée aussi *contrainte*, est obligatoire dans la mesure où un élément ne satisfaisant pas cette condition est indésirable et définitivement rejeté. La condition positive est, quant à elle, optionnelle dans la mesure où un élément ne vérifiant pas cette condition n'est pas nécessairement rejeté.

Dans [13, 40], les auteurs définissent la condition négative comme étant l'information indiquant ce qui doit être rejeté et éliminé et, la condition positive comme l'information déterminant ce qui est réellement souhaité. Quant aux travaux [48, 152], la condition négative est considérée comme celle spécifiant, par complémentation, la contrainte (c) décrivant les éléments acceptables et, la condition positive représente le souhait (w) utilisé pour différencier entre les éléments préférés parmi ceux qui vérifient la contrainte (c). Dans [48], la propriété de consistance ou de cohérence doit être vérifiée pour chaque couple (c,w), i.e., $\forall x, \mu_c(x) \leq \mu_c(x)$, où $\mu_c(x)$ et $\mu_w(x)$ correspondent respectivement au degré de satisfaction de la contrainte (c) et du souhait (w) de l'élément $x$. En d'autres termes, cette règle signifie qu'on ne peut préférer ce qu'on rejette [148] (i.e., $\{x|x$ est un élément qui satisfait w$\} \subseteq \{x|x$ est un élément qui satisfait c$\}$). Par ailleurs, dans leur définition d'une requête bipolaire, Zadrożny et Kacprzyk [150, 151] n'imposent aucune condition de cohérence entre les contraintes et les souhaits.

## 2.3 Approches agrégatives

Les préférences dans les requêtes utilisateur peuvent, d'une part, éviter les réponses vides dues aux contraintes très sélectives de l'utilisateur, et d'autre part, fournissent un ensemble adéquat de résultats pertinents, même lorsque l'utilisateur spécifie des contraintes assez générales.

La théorie des ensembles flous [148] offre des outils efficaces pour exprimer et évaluer des requêtes intégrant de telles préférences. Par conséquent, des approches basées sur les ensembles flous ont été proposées, pour définir des préférences, en utilisant des adjectifs et/ou adverbes du langage naturel, comme *jeune, élevé*, *proche*, *la plupart*, etc. Ces derniers sont appelés « ***prédicats graduels*** » ou encore « ***prédicats flous*** », comme défini dans la section 2.2.

Les ensembles flous pour l'évaluation des requêtes à préférences, permettent de bien interpréter les termes linguistiques que l'utilisateur peut incorporer dans sa requête, mais aussi de combiner plusieurs préférences de différents attributs (l'hypothèse de commensurabilité), pour obtenir au final un pré-ordre total sur les réponses.

Dans cette section, nous présentons les principales méthodes basées sur la théorie des ensembles flous pour modéliser et évaluer les préférences utilisateur, en combinant les degrés de satisfaction des conditions élémentaires de même type. La sous-section 2.3.1 résume les approches de modélisation et d'évaluation de requêtes à préférences unipolaires, et les modèles de préférences de type bipolaire sont résumés en sous-section 2.3.2.

### 2.3.1 Approches unipolaires

Plusieurs approches ont été proposées pour évaluer les requêtes à préférences composées de conditions élémentaires de même type, i.e., préférences unipolaires. La modélisation de ces conditions à l'aide de fonctions d'appartenance permet d'agréger leurs degrés de satisfaction pour déterminer un seul degré global. Ainsi, les éléments évalués peuvent être ordonnés des plus pertinents aux moins pertinents vis-à-vis de l'utilisateur. Parmi ces méthodes, nous présentons dans la suite de cette sous-section les travaux [16, 15, 45, 125, 126, 142].

#### 2.3.1.1 Approche de Dubois et Prade

Afin de répondre aux requêtes flexibles à préférences unipolaires, une approche basée sur la théorie des ensembles flous est présentée dans [45]. Ces préférences sont évaluées à l'aide des fonctions d'appartenance correspondant aux prédicats flous présents dans la requête de l'utilisateur. Comme illustré dans l'exemple précédent, une préférence peut être composée de plusieurs conditions élémentaires sur différents attributs. Pour évaluer de telles préférences, un des opérateurs d'agrégation de la théorie des ensembles flous, peut être appliqué. À titre d'exemple, les opérateurs « min » et « max » peuvent être utilisés pour exprimer, respectivement, des conditions conjonctives et des conditions disjonctives.

Les auteurs proposent une nouvelle expression permettant de calculer les degrés de satisfaction des préférences en prenant en considération leur niveau d'importance. Pour cela, la formule suivante

est appliquée :

$$\mu'_i = max(\mu_i, 1 - \omega_i) \tag{2.1}$$

où $\omega_i$ dénote l'importance d'une condition $c_i$ sur l'attribut $a_i$ présent dans la requête utilisateur, avec $\max_i \omega_i = 1$ (il existe toujours une condition d'importance maximale) et $\mu_i$ représente le degré d'appartenance (ou de satisfaction) d'un élément $x$ par rapport à la condition $c_i$.

La formule (2.1) nous permet de calculer le degré de satisfaction des conditions élémentaires (i.e. les préférences atomiques) de l'utilisateur en tenant compte non seulement de leurs prédicats flous mais aussi de leur niveau d'importance. Elle exprime que plus $\omega_i$ est grand, plus la satisfaction d'une condition élémentaire $c_i$ (i.e., $\mu_i$) est prise en compte. En particulier, une valeur en dehors du support de la fonction d'appartenance associée à une condition $c_i$, est satisfaisante à un degré égal à $1-\omega_i$. Cela signifie que plus grand est $\omega_i$ (i.e., $c_i$ est importante), plus petit est le degré de satisfaction d'une valeur en dehors du support de $c_i$.

Dubois et Prade [45] proposent l'application des agrégations conjonctives ou disjonctives pondérées, des conditions $(c_1, c_2, ..., c_n)$, d'une requête utilisateur $q$ avec préférence, en utilisant respectivement les deux expressions (2.2) et (2.3). La formule (2.2) évalue donc le degré de satisfaction global d'un élément $x$ par rapport à une requête $q$ ayant des conditions de la forme « $c_1$ **et** ... **et** $c_n$ ». De la même manière, pour un élément $x$, la formule (2.3) détermine, elle aussi, la satisfaction globale d'une requête$q$, mais cette fois-ci composée d'un ensemble de conditions sous la forme « $c_1$ **ou** ... **ou** $c_n$ ».

$$\mu_q = \min_{i=1,...,n} \max(\mu_i, 1 - \omega_i) \tag{2.2}$$

et

$$\mu_q = \max_{i=1,...,n} \min(\mu_i, \omega_i) \tag{2.3}$$

où $\omega_i$ représente l'importance d'une condition $c_i$ sur l'attribut $a_i$ (parmi les $n$ attributs) présent dans la requête utilisateur $q$, avec $\max_i \omega_i = 1$ (car il existe toujours une condition d'une importance maximale), $\mu_i$ représente le degré de satisfaction d'un élément $x$ par rapport à la condition $c_i$ et $\mu_q$ dénote le degré de satisfaction de l'élément $x$ par rapport à toutes les conditions $c_i$ de la requête $q$.

Par exemple, pour la requête « *trouver un hôtel pas cher et aussi proche que possible de la plage* », une agrégation avec l'opérateur « min » (i.e., la formule (2.2)) peut être appliquée sur les deux degrés d'appartenance des prédicats « pas cher » et « proche ».

#### 2.3.1.2 Intégrales floues

Un des rôles d'un processus d'évaluation de requêtes contenant des préférences exprimées sur plusieurs critères (attributs ou dimensions), est d'ordonner les réponses retournées en fonction de ces préférences. Pour cela, une méthode d'agrégation est possible, pour calculer un seul score à partir

des différents critères exprimés dans la requête.

En effet, divers opérateurs d'agrégation existent, comme la moyenne arithmétique ou pondérée, la combinaison pondérée de « min » et de « max », OWA (Ordered Weighted Average), ..., mais aucun d'entre eux ne prend en compte *la puissance de la coalition* entre les critères, i.e. l'importance des sous-ensembles de critères.

Nous pouvons voir l'importance de la notion de « *coalition* », par exemple, dans le cas d'une personne qui veut acheter une voiture en fonction de trois critères de sélection : *la puissance*, *le nombre de sièges* et *le prix*. Sa requête peut donc être de la forme « *sélectionner une voiture pas chère ayant une puissance élevée et un nombre de sièges* $\geq 4$ ». Selon ses préférences, chacun de ces trois critères a un poids d'importance, par exemple, 0.33 pour le critère « *puissance* », 0.66 pour « *nombre de sièges* » et 0.5 pour le critère « *prix* ». Supposons maintenant que cette personne soit un père de famille. Du point de vue de cet utilisateur, l'importance de coalition de « *nombre de sièges* » et « *prix* » est nécessairement plus élevée que toute autre coalition de cardinalité égale à 1 ou 2.

Les intégrales floues sont l'une des méthodes utilisées, comme fonctions d'agrégation, pour l'évaluation de telles requêtes et prendre en considération l'importance des sous-ensembles de critères. Elles sont très utiles dans de nombreuses applications, comme dans le domaine de reconnaissance de formes, la médecine, ... En effet, elles permettent de déterminer la meilleure combinaison de valeurs des critères, pour l'utiliser ensuite dans la prise de décision finale.

Les intégrales floues les plus utilisées sont l'intégrale floue de Sugeno, pour une évaluation qualitative [125, 126], et l'intégrale floue de Choquet, pour une évaluation quantitative [31]. Ces dernières sont des fonctions définies à l'aide de leur mesure floue.

Dans cette sous-section, nous présentons d'abord les mesures floues et leur détermination. Ensuite, nous présentons l'intégrale floue de Sugeno. Enfin, nous terminons la sous-section en définissant l'intégrale de Choquet.

Considérons, dans ce qui suit, $X = \{x_1, ..., x_n\}$ un ensemble fini et discret de $n$ éléments.

#### 2.3.1.2.1 Mesures floues

Les *mesures floues*, dites aussi *capacités*, permettent de déterminer l'importance combinée, la fiabilité ou encore la satisfaction d'un sous-ensemble de critères. Une mesure floue sur un ensemble fini $X$ est une fonction ensembliste monotone définie par :

$$g : P(X) \rightarrow [0, 1], \tag{2.4}$$

vérifiant les axiomes suivants :

- $g(\emptyset) = 0$, $g(X) = 1$ et
- pour tout $S, T \subseteq X$, si $S \subseteq T$ alors $g(S) \leq g(T)$ (propriété de monotonie).

où $P(X)$ représente l'ensemble des sous-ensembles de $X$ et, $g(S)$ et $g(T)$ dénotent, respectivement, la mesure du degré auquel le sous-ensemble $S$ et le sous-ensemble $T$ satisfont le concept (i.e. l'importance ou la satisfaction) mesuré par $g$. En d'autres termes, $g(S)$ et $g(T)$ sont, respectivement,

l'importance ou la puissance de coalition du sous-ensemble $S$ et celle du sous-ensemble $T$.

#### 2.3.1.2.2 Détermination des mesures floues

La principale difficulté des intégrales floues réside dans l'identification de leurs mesures, à savoir l'identification de $2^n - 2$ valeurs d'une mesure floue $g$, où $n$ est le nombre de critères à regrouper, soit par des données d'apprentissage ou par un questionnaire ou les deux à la fois.

Rappelons que, contrairement aux mesures floues, les autres opérateurs d'agrégation, tels que les opérateurs de conjonction ou de disjonction, les opérateurs arithmétiques ou moyenne pondérée, OWA, ..., définissent seulement le poids d'importance de chaque élément (singleton) de $X$. Mais, comme la mesure floue de l'union de deux sous-ensembles disjoints ne peut pas toujours être directement calculée à partir des mesures floues des sous-ensembles, plusieurs travaux [63, 83, 27, 32, 123, 136, 129] ont été réalisés pour calculer toutes les mesures floues.

Par exemple, Sugeno a défini une mesure décomposable à partir des mesures de singleton [126], dite « $\lambda$-fuzzy measure », qui satisfait la propriété supplémentaire suivante :

$$g\left(S \cup T\right) = g\left(S\right) + g\left(T\right) + \lambda g\left(S\right).g\left(T\right) \tag{2.5}$$

pour tout $S, T \subseteq X$ , $S \cap T = \emptyset$ et pour une valeur donnée de $\lambda \geq -1$.

Par conséquent, pour déterminer les mesures floues suivant la fonction $g$, la valeur de $\lambda$ doit être calculée en résolvant l'équation (2.6) ou (2.7), permettant de trouver une valeur unique de $\lambda$ supérieure à $-1$. Voir [127], pour plus de détails sur la procédure standard calculant la valeur de $\lambda$.

$$\lambda + 1 = \prod_{i=1}^{n} \left(1 + \lambda g\left(\{x_i\}\right)\right) \tag{2.6}$$

$$g\left(S\right) = \left[\prod_{x \in S} \left(1 + \lambda g\left(\{x\}\right)\right)\right] / \lambda \tag{2.7}$$

En conséquence, les mesures floues peuvent être déterminées de deux manières différentes :

1. une méthode *séquentielle* basée sur la formule (2.5), qui consiste à énumérer $2^n - 2$ mesures floues, parmi lesquelles seulement $n$ mesures seront utilisées dans le calcul de l'intégral floue de Sugeno, ou
2. une méthode *récursive* en utilisant les formules (2.8) et (2.9). Contrairement à la première méthode, cette dernière évalue uniquement les $n$ mesures floues nécessaires pour calculer l'intégrale floue de Sugeno de $n$ critères, ce qui réduit la complexité de la méthode.

$$g\left(A_1\right) = g\left(x_1\right) \tag{2.8}$$

$$g\left(A_i\right) = g\left(\{x_i\}\right) + g\left(A_{i-1}\right) + \lambda . g\left(\{x_i\}\right).g\left(A_{i-1}\right) \tag{2.9}$$

où $i = 1, ..., n$, $g(\{x_i\})$ correspond aux mesures floues des sous-ensembles de $X$ composés d'un seul élément $x_i$. Ces mesures sont déterminées par l'utilisateur ou par un expert et classées par ordre

croissant suivant les degrés à combiner (i.e. pour toutes les mesures floues $g(\{x_i\})$ et $g(\{x_{i+1}\})$ : $\mu(x_i) \leq \mu(x_{i+1})$). Cependant, le calcul de la valeur de $\lambda$ n'est toujours pas facile.

#### 2.3.1.2.3 Intégrale floue de Sugeno

L'intégrale floue de Sugeno [125] peut être utilisée comme une fonction d'agrégation de $[0,1]^n$ dans $[0,1]$, permettant d'effectuer une évaluation *qualitative* d'un ensemble de critères.

Soit $\mu : X \rightarrow [0,1]$ une fonction de sous-ensembles de $X = \{x_1, ...x_n\}$. L'intégrale floue de Sugeno, notée $S$, suivant la fonction d'appartenance $\mu$ et la mesure floue $g$, est définie par :

$$S(X, \mu, g) = \max_{i=1,...,n} (\min(\mu(x_i), g(A_i))) \tag{2.10}$$

où $\mu(x_1) \leq \ldots \leq \mu(x_n)$ et $A_i = \{x_i, \ldots, x_n\}$.

L'interprétation de l'intégrale de Sugeno peut être définie comme suit : $\mu(x_i)$ exprime le degré avec lequel le concept représenté par $\mu$ est satisfait par le critère $x_i$. $g(A_i)$ représente, quant à lui, le degré avec lequel le sous-ensemble d'éléments $A_i = \{x_i, ..., x_n\}$ satisfait le concept mesuré par la fonction $g$. Ainsi, $\min(\mu(x_i), g(A_i))$ calcule le degré avec lequel le concept représenté par $\mu$ est satisfait par *tous* les éléments de $A_i = \{x_i, ..., x_n\}$. Autrement dit, le concept $\mu$ est satisfait par tous les éléments de $A_i = \{x_i, ..., x_n\}$ à au moins un degré égal à $\min(\mu(x_i), g(\{x_i, ..., x_n\}))$. Enfin, la plus grande valeur (i.e. la meilleure combinaison) est choisie par l'opérateur « max ».

Plusieurs travaux ont été réalisés [65, 64, 100, 44] pour étudier les caractéristiques de l'intégrale de Sugeno comme une fonction d'agrégation. Il a été démontré que :

- l'intégrale de Sugeno contient les statistiques d'ordre, telles que le min, le max et la médiane ;
- les minimum et maximum pondérés et les minimum et maximum ordonnés pondérés sont des cas particuliers de l'intégrale de Sugeno.

De plus, il est important de noter que, l'intégrale de Sugeno est un opérateur *ordinal qualitatif*. En effet, l'ordre des critères a un rôle important dans le processus d'agrégation multicritères.

#### 2.3.1.2.4 Intégrale floue de Choquet

L'intégrale de Choquet [106], notée $C$, est un opérateur *quantitatif*, dont son calcul consiste à utiliser la mesure floue de la même manière qu'une mesure additive, c'est-à-dire :

$$C(X, \mu, g) = \sum_{i=1}^{n} \mu(x_i).(g(A_i) - g(A_{i+1}))$$

où $\mu(x_1) \leq \ldots \leq \mu(x_n)$, $A_i = \{x_i, \ldots, x_n\}$, $A_{n+1} = \emptyset$, $\mu : X \rightarrow [0,1]$ est une fonction de sous-ensembles de $X$ et $g$ est une mesure floue.

Ainsi, l'intégrale de Choquet peut être interprétée comme suit : $\mu(x_i)$ exprime le degré d'importance du critère $x_i$ dans l'ensemble $X$ suivant le concept représenté par $\mu$. $g(A_i)$ détermine le degré d'importance du sous-ensemble d'éléments $A_i = \{x_i, ..., x_n\}$ par rapport au concept mesuré par la fonction $g$.

Cette intégrale floue, généralise elle aussi comme l'intégrale de Sugeno, plusieurs opérateurs d'agrégation classiques [64], tels que la moyenne arithmétique pondérée ou les opérateurs OWA :

- lorsque la mesure floue, $g$, est additive, c'est-à-dire $g(A) = \sum_{x_i \in A} g(\{x_i\})$ pour tout $A \subseteq X$, l'intégrale de Choquet coïncide avec la somme pondérée ayant $\omega_i = g(\{x_i\})$, et
- la moyenne pondérée ordonnée (OWA) correspond aussi à l'intégrale de Choquet, lorsque la mesure floue, $g$, est symétrique, c'est-à-dire vérifiant $g(A) = g(B)$ si $|A| = |B|$.

Toutefois, l'intégrale de Choquet contrairement à l'intégrale de Sugeno, est adaptée pour une agrégation *cardinale* (donc une évaluation *quantitative*), car seul le nombre d'éléments satisfaisants, a un sens réel dans le processus d'évaluation.

Dans [66], les auteurs proposent le schéma illustré dans la figure 2.2, permettant de résumer les différentes relations entre les opérateurs d'agrégation classiques et les intégrales floues.

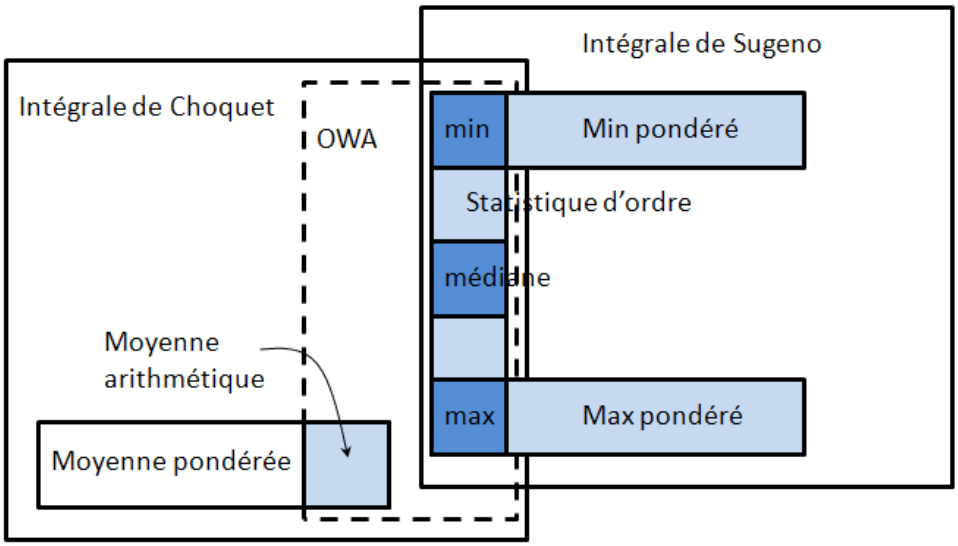



FIGURE 2.2 – Relations entre les opérateurs d'agrégation et les intégrales floues.

#### 2.3.1.3 Approche basée sur les quantificateurs linguistiques

Une requête à préférences peut être aussi exprimée avec *une proposition quantifiée*, en utilisant un ou plusieurs quantificateurs linguistiques, tels que *presque tous*, *la plupart*, *la moitié*, ..., sur les conditions la composant. À titre d'exemple, un utilisateur peut *rechercher les services Web satisfaisant* ***presque toutes*** *les conditions : être rapide, être pas cher et être fiable*. Ces quantificateurs permettent ainsi de combiner et d'agréger les conditions élémentaires d'une requête pour l'évaluation de cette dernière. Ils peuvent être flous comme dans la cas de la requête précédente, ou numériques comme dans *« rechercher les maisons satisfaisant* **80%** *des conditions : être proche de la plage, être pas cher et avoir une surface d'environ* $250\text{m}^2$ *»*.

Les quantificateurs flous ont été introduits par Zadeh [149], généralisant les quantificateurs universel ($\forall$) et existentiel ($\exists$). Les expressions : au moins 3, environ la moitié, presque tout, ..., sont des exemples de tels quantificateurs. Ainsi, on distingue deux types de quantificateurs :

1. *les quantificateurs absolus*, exprimant une quantité ou un nombre absolu comme environ 4 ou au moins 3, définis par une fonction $\mu_Q$ telle que :

$$\begin{aligned} \mu_Q : \mathbb{R}(\text{ ou } \mathbb{N}) &\rightarrow [0,1] \\ x &\mapsto \mu_Q(x) \end{aligned}$$

2. et *les quantificateurs relatifs*, spécifiant une proportion comme presque tous, au moins la moitié ou encore environ un quart, modélisés par une fonction $\mu_Q$ telle que :

$$\begin{aligned} \mu_Q : [0,1] &\rightarrow [0,1] \\ x &\mapsto \mu_Q(x) \end{aligned}$$

La valeur $\mu_Q(x)$, dans les fonctions ci-dessus, exprime respectivement la satisfaction du quantificateur $Q$ suivant un nombre ou un proposition $x$. Ceci traduit donc l'adéquation du nombre ou de la proportion $x$ avec le quantificateur $Q$.

Un quantificateur linguistique peut être aussi : *croissant* (tel que presque tous, au moins 3, au moins la moitié, ...), *décroissant* (comme au plus 3 ou au plus un sixième) ou *unimodal* (comme environ 5 ou environ la moitié). Un quantificateur est dit *monotone* s'il est croissant ou décroissant (voir [95, 61]). La figure 2.3 illustre des exemples définissant, à l'aide de fonctions d'appartenance, des quantificateurs linguistiques croissant, décroissant et unimodal.

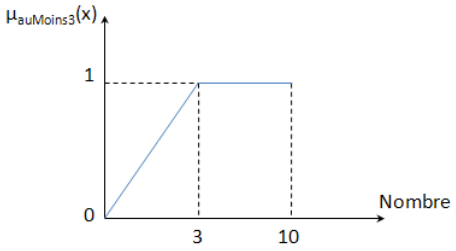
#### 2.3.1.3.1 Propositions quantifiées

À l'aide des quantificateurs linguistiques, les expressions dites quantifiées peuvent être évaluées, en agrégeant l'ensemble des valeurs représentant les degrés de satisfaction des conditions élémentaires les constituant. Ces expressions sont de la forme « Q X sont P » ou « Q B X sont P », où Q est le quantificateur, X est un ensemble d'éléments et, B et P sont deux prédicats flous.

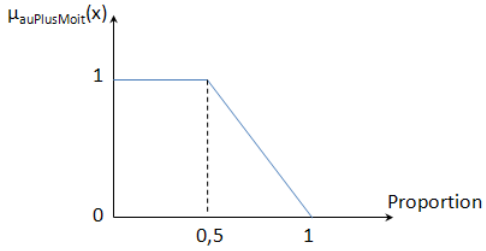
Les propositions de type « Q X sont P », peuvent se lire comme « parmi les éléments de X, il y en a Q qui satisfont P ». Un exemple de telles propositions est « trouver une entreprise de construction dont *la plupart* des services sont *pas chers* », avec Q le quantificateur linguistique « *la plupart* », X l'ensemble des services d'une entreprise de construction et P le prédicat flou « *pas chères* ».

Quant aux propositions de type « Q B X sont P », elles s'interprètent comme suit : « parmi les éléments de X qui satisfont B, il y en a Q qui satisfont P ». La proposition « trouver une entreprise de construction dont *la plupart* des services *pas chers* sont *de bonne qualité* » est une expression quantifiée de type « Q B X sont P », avec Q le quantificateur linguistique « *la plupart* », X l'ensemble des services d'une entreprise de construction et, B et P les prédicats flous respectifs « *pas chers* » et « *une bonne qualité* ».

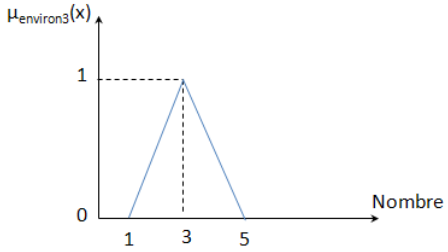
Étant les seules considérées dans la suite de cette thèse, les propositions de la forme « Q X sont P » seront les seules présentées dans cette section. Voir [95] pour plus de détails sur les expressions




(a) Exemple de définition du quantificateur croissant « au moins 3 ».




(b) Exemple de définition du quantificateur décroissant « au plus la moitié ».




(c) Exemple de définition du quantificateur unimodal « environ 3 ».

FIGURE 2.3 – Exemples de quantificateurs linguistiques.

quantifiées de type « Q B X sont P ».

Afin de répondre aux requêtes contenant des expressions quantifiées de type « Q X sont P », plusieurs méthodes ont été proposées [95, 125, 142, 143]. Les propositions « Q B X sont P » sont difficiles à évaluer car un problème d'interprétation se pose lorsque l'ensemble des éléments vérifiant le prédicat « B » est vide (ou presque vide). Ainsi, quelle signification donne t-on à l'expression « la plupart des jeunes employés sont bien payés », lorsque l'ensemble des jeunes est vide ? Pour une étude plus détaillée et plus complète, voir [95].

L'objectif de ces approches est donc de calculer ***dans quelle mesure*** *Q éléments de X satisfont le prédicat flou P*. Dans le cas d'un quantificateur absolu, la mesure dans laquelle le nombre d'éléments de l'ensemble X vérifiant le prédicat P est compatible avec le quantificateur Q, est calculée. Quant à l'utilisation d'un quantificateur relatif dans une expression « Q X sont P », elle permet d'évaluer à quel degré la proportion des éléments de X satisfaisant P est compatible avec le quantificateur Q.

Dans cette sous-section, nous nous restreignons à présenter l'approche par décomposition de Yager [142], qui est non seulement satisfaisante en terme de propriétés mais aussi en terme de compatibilité avec les mesures floues et l'interrogation flexible des bases de données. Cette approche est la seule considérée dans la suite de cette thèse pour évaluer les expressions quantifiées.

#### 2.3.1.3.2 Approche par décomposition

Dans [142], Yager a proposé une méthode permettant d'interpréter et d'évaluer les propositions quantifiées exprimées avec des quantificateurs croissants, comme la plupart, au moins 2, presque tous, au moins un quart, ... Cette approche est composée des étapes suivantes :

– Soit $\Omega = \{\mu_1, ..., \mu_n\}$ un ensemble de degrés de satisfaction des éléments de X par rapport au prédicat P, ordonnés dans l'ordre décroissant ; i.e. $\mu_1 \geq ... \geq \mu_n$ ;
– Le degré de vérité, noté $\delta$, évaluant l'expression « Q X sont P » est calculé par l'une des deux formules (2.11) et (2.12) :

1) Q quantificateur absolu :

$$\delta = \max_{i=1,...,n} \min\left(\mu_i, \mu_Q\left(i\right)\right) \tag{2.11}$$

2) Q quantificateur relatif :

$$\delta = \max_{i=1,...,n} \min\left(\mu_i, \mu_Q\left(i/n\right)\right) \tag{2.12}$$

où $\mu_Q\left(i\right)$ (resp., $\mu_Q\left(i/n\right)$) est la valeur de vérité de l'expression quantifiée quand exactement i (resp., $i/n$) éléments de X satisfont totalement le prédicat P.

Les deux formules (2.11) et (2.12) permettent de trouver un meilleur compromis entre la quantité exprimée par le quantificateur « Q » et la satisfaction du prédicat « P » et peuvent s'écrire également comme suit :

$$\delta = \min_{i=1,...,n} \max\left(\mu_i, 1 - \lambda_i\right) \tag{2.13}$$

où pour un quantificateur absolu Q, $\lambda_i = 1 - \mu_Q\left(i-1\right)$ et, pour un quantificateur relatif Q, $\lambda_i = 1 - \mu_Q\left((i-1)/n\right)$.

#### 2.3.1.4 SQLF

SQLF, proposé par Bosc *et al.* [16, 15], est une extension de SQL permettant l'interrogation flexible des bases de données. Ce langage permet d'évaluer des requêtes graduelles au moyen de la théorie des ensembles flous. Ainsi, chaque réponse est associée à degré exprimant son degré de satisfaction par rapport à une requête SQLF. Dans ce qui suit, nous présentons quelques éléments importants de ce langage.

SQLF a été défini d'une manière à préserver les trois clauses « **SELECT** », « **FROM** » et « **WHERE** » du bloc de base de SQL. Rappelons que la structure de base d'une requête SQL est de la forme suivante :

| | |
|---|---|
| **SELECT** | < informations à sélectionner > |
| **FROM** | < listes des relations interrogées > |
| **WHERE** | < conditions> |

La clause « SELECT » détermine les attributs constituant les réponses de la requête, « FROM » spécifie la liste des relations interrogées et la clause « WHERE » définit les conditions que doivent respecter les réponses retournées.

Dans une requête SQLF, la clause « FROM » est identique à celle du langage SQL et les différences concernent principalement deux aspects :

- **le calibrage du résultat** : contrairement aux requêtes SQL, les requêtes SQLF peuvent contenir deux informations supplémentaires : (i) le nombre « n » de réponses souhaitées et/ou (ii) un seuil « $\alpha$ » permettant la sélection des réponses ayant un degré de satisfaction supérieur ou égal à $\alpha$ ;
- **la nature des conditions autorisées** : la clause « WHERE » d'une requête SQLF peut contenir des conditions floues et/ou booléennes.

Ainsi, la structure du bloc de base en SQLF est de la forme :

**SELECT** < informations à sélectionner >
**FROM** < listes des relations interrogées >
**WHERE** < conditions floues et/ou booléennes>
**[ LIMIT** n ]
**[ ALPHACUT** t ]

où les clauses entre crochets sont optionnelles. L'évaluation de ce type de requêtes consiste en la sélection floue du produit cartésien des relations appartenant à la clause « FROM », suivie d'une projection sur les attributs de « SELECT », puis d'un calibrage des résultats pour déterminer les « n » meilleures réponses et/ou ceux qui ont un degré de satisfaction supérieur ou égal à $\alpha$.

**Exemple 2.3.1.** Soit la requête : « rechercher les hôtels *proches* de la plage, avec un degré de satisfaction supérieur à 0,8 ». Cette dernière s'écrit en SQLF comme suit :

**SELECT** n-Hôtel, nom-Hôtel
**FROM** Hôtel
**WHERE** Hôtel.Dist **is** proche
**ALPHACUT** 0,8

L'évaluation de cette requête retourne une relation floue composée des attributs « *n-Hôtel* » et « *nom-Hôtel* ». Les degrés d'appartenance à cette relation sont les degrés de satisfaction de chaque hôtel à la condition « *proche* de la plage et degré supérieur à 0,8 ». □

L'utilisation de la théorie des ensembles flous dans SQLF a un grand avantage dans l'interrogation des bases de données, car elle offre une panoplie d'opérateurs, tels que « min » et « max » (éventuellement pondérés), permettant l'expression de prédicats complexes. Il est également possible d'utiliser un quantificateur flou portant sur un ensemble de conditions pour définir une attitude intermédiaire entre la conjonction et la disjonction de conditions floues [95].

**Exemple 2.3.2.** Soit la requête : « rechercher les hôtels satisfaisant la plupart des conditions : avoir au moins trois étoiles, pas chers, proches de la plage et avoir un très bon restaurant ». Cette dernière s'écrit en SQLF suit :

**SELECT** n-Hôtel, nom-Hôtel
**FROM** Hôtel
**WHERE** la-plupart(Hôtel.Étoil $\geq 3$
**AND** Hôtel.prix **is** PasCher

**AND** Hôtel.Dist **is** proche
**AND** Hôtel.Rest **is** bon)

Cette requête correspond à une proposition quantifiée floue permettant de trouver un compromis entre la satisfaction de chaque condition et celle du quantificateur. □

L'évaluation de ce type de requêtes peut être réalisée en utilisant une intégrale floue de Sugeno [125] (définie dans la sous-section 2.3.1.2) ou bien en appliquant la méthode par décomposition de Yager [142] (présentée en sous-section 2.3.1.3).

SQLF permet également d'utiliser des prédicats construits par imbrication d'un autre prédicat appelé « sous-requête ». Les opérateurs permettant la définition d'une sous-requête sont l'appartenance (in), la « non-vacuité » (exists), la comparaison à une ou plusieurs valeurs retournées ($\theta$ any et $\theta$ all) et leur négation (not).

Enfin, le langage SQLF permet d'effectuer des partitionnements de relations. Il s'agit de regrouper des n-uplets en utilisant la clause « group by » et d'exprimer une condition ensembliste floue sur le regroupement par la clause « having ».

### 2.3.2 Approches bipolaires

De nombreux modèles d'expression et d'évaluation de requêtes bipolaires, ont été proposés. Nous présentons dans la suite de cette sous-section trois approches : l'approche de Dubois et Prade, l'approche hiérarchique et celle basée sur l'opérateur « winnow » [47, 46, 98, 97, 130, 150, 151]. Voir en particulier [131], pour une étude plus complète et plus détaillée des approches d'évaluation des requêtes bipolaires.

#### 2.3.2.1 Approche de Dubois et Prade

Dans [47, 46], Dubois et Prade considèrent une requête bipolaire comme étant une association des deux conditions, négative et positive, dans laquelle la condition négative représente des contraintes et la condition positive exprime des souhaits. Les deux conditions considérées dans cette approche, sont celles formulées à l'aide de prédicats flous (voir la sous-section 2.2 page 42), car ces dernières généralisent le cas des conditions booléennes.

Pour répondre à de telles requêtes (floues bipolaires), l'ensemble des tuples ne satisfaisant pas la contrainte « $c$ » sont complètement rejetés et, les tuples ayant la même satisfaction au niveau de la contrainte, sont départagés suivant leur satisfaction par rapport au souhait « $w$ » de l'utilisateur. En effet, pour une même satisfaction de la contrainte, plus un tuple satisfait le souhait, plus il est préféré.

Comme un tuple ne peut pas être plus souhaité qu'il n'est acceptable par rapport à la contrainte, l'évaluation des requêtes bipolaires doit respecter *la propriété* dite *de consistance* ou *de cohérence*. Pour cela, chaque condition bipolaire, notée $(c, w)$, de la forme « *contrainte "c" et si possible souhait "w"* », doit être réécrite en « *contrainte "c" et si possible contrainte "c" et souhait "w"* ». La condition de cohérence devient donc : $\forall x \in X, \mu_w(x) \leq \mu_c(x)$, où $\mu_w(x)$ (resp., $\mu_c(x)$) dénote le degré de satisfaction de l'élément (ou tuple) $x$ par rapport au souhait $w$ (resp., à la contrainte $c$).

La satisfaction d'une requête bipolaire composée d'une contrainte "$c$" et d'un souhait "$w$", est représentée par un couple de degrés dans $[0,1]\times[0,1]$ et, chaque élément $x \in X$ possède une paire de degrés de satisfaction, notée $(\mu_c(x), \mu_w(x))$. Si aucune contrainte n'est exprimée, alors aucun élément $x$ ne sera rejeté et $\forall x \in X, \mu_c(x) = 1$ et, si aucun souhait n'est défini alors $\forall x \in X, \mu_w(x) = 0$. Pour assurer la propriété de cohérence, la paire $(\mu_c(x), \mu_x(x))$ d'une condition bipolaire peut être remplacée par : $(\mu_c(x), \min((\mu_c(x), \mu_w(x)))$.

Dans le cas d'une requête bipolaire complexe, c'est-à-dire une conjonction de "$n$" conditions bipolaires de la forme $(c_1, w_1) \wedge (c_2, w_2) \wedge ... \wedge (c_n, w_n)$, Dubois et Prade combinent les paires pour obtenir deux degrés globaux, notés $(\mu_c(x), \mu_w(x))$, représentant, respectivement, la satisfaction des contraintes et celle des souhaits. Pour cela, la formule suivante est appliquée :

$$(c(x), w(x)) = (\min_{i=1,...,n} \mu_{c_i}(x), \max_{i=1,...,n} \mu_{w_i}(x)) \tag{2.14}$$

où $\mu_{c_i}(x)$ (resp., $\mu_{w_i}(x)$) représente le degré de satisfaction de l'élément $x$ suivant la contrainte $c_i$ (resp., le souhait $w_i$), pour $i = 1, ..., n$.

La formule (2.14) permet d'exprimer dans quelle mesure l'élément $x$ satisfait *toutes les contraintes et si possible au moins un souhait*. Pour garantir une cohérence dans l'évaluation, cette formule est donc réécrite comme suit :

$$(c(x), w(x)) = (\min_{i=1,...,n} \mu_{c_i}(x), \min(\min_{i=1,...,n} \mu_{c_i}(x), \max_{i=1,...,n} \mu_{w_i}(x))) \tag{2.15}$$

Par exemple, la requête « *rechercher les hôtels trois étoiles, pas chers et si possible proche de la plage et ayant un très bon restaurant* », est une requête bipolaire complexe composée de deux contraintes ($c_1$ et $c_2$) et de deux souhaits ($w_1$ et $w_2$) exprimées, respectivement, sur les attributs : nombre d'étoiles, prix, distance par rapport à la plage et qualité du restaurant. Autrement dit, cette requête est une composition des trois conditions bipolaires suivantes : $(c_{TroisEtoiles}, 0)$, $(c_{PasCher}, w_{Proche})$ et $(1, w_{TrèsBon})$.

Pour évaluer chaque élément $x$ de la base de données par rapport à cette requête, les degrés de satisfaction des contraintes et des souhaits sont calculés à l'aide de la théorie des ensembles flous. Soient $\mu_{TroisEtoiles}(x.nbEtoile)$, $\mu_{PasCher}(x.Prix)$, $\mu_{Proche}(x.Distance)$ et $\mu_{TrèsBon}(x.QualitéRest)$ les degrés de satisfaction de $c_1$, $c_2$, $w_1$ et $w_2$, respectivement, associés à l'élément $x$. Le degré de satisfaction global de ce dernier par rapport aux contraintes, est déterminé en appliquant l'opérateur « min » comme suit : $\mu_c(x) = \min(\mu_{TroisEtoiles}(x.nbEtoile),\ \mu_{PasCher}(x.Prix))$. Quant à son degré global par rapport aux souhaits, il est calculé en utilisant l'opérateur « max », c.-à-d., $\mu_w(x) = \max(\mu_{Proche}(x.Distance), \mu_{TrèsBon}(x.QualitéRest))$. Par conséquent, l'évaluation de $x$ par rapport à la requête précédente est représentée par le couple de degrés $(c(x), w(x)) = (\mu_c(x), \mu_w(x))$.

– **Classement des réponses**

En appliquant l'approche de Dubois et Prade [47, 46] décrite ci-dessus, l'évaluation de chaque élément $x$ suivant une requête bipolaire floue, permet donc d'obtenir à l'aide de la formule (2.15),

un couple de degrés $(c(x), w(x))$. Pour retourner l'ensemble $\{x_1, x_2, ..., x_m\}$ des éléments réponses à la requête, il est toutefois important d'appliquer une méthode permettant de les ordonner du plus satisfaisant au moins satisfaisant.

**1. Classement sans agrégation**

Pour l'ordonnancement des éléments réponses, un ordre lexicographique peut être utilisé en triant d'abord les éléments sur les valeurs $c(x_i)$, ensuite sur les valeurs $w(x_i)$ pour départager les ex-æquo au niveau de la contrainte. Autrement dit, un élément $x_1$ est plus satisfaisant que (est préféré à) l'élément $x_2$, noté $x_1 \succeq x_2$, signifie que : $(c(x_1) > c(x_2)) \vee ((c(x_1) = c(x_2)) \wedge (w(x_1) \geq w(x_2)))$. Par exemple, pour une requête bipolaire floue donnée, ses éléments réponses $x_1(0.4, 0.6)$, $x_2(0.9, 0.1)$ et $x_3(0.4, 0.8)$ sont ordonnés comme suit : $x_2 \succeq x_3 \succeq x_1$. En effet, l'élément $x_2$ est préféré à $x_1$ et à $x_3$ car c'est le plus satisfaisant au niveau de la contrainte $c$ et, $x_3$ est préféré à $x_1$ car ils satisfont au même degré la contrainte $c$ mais $x_2$ est meilleur au niveau du souhait $w$.

**2. Classement basé sur l'agrégation**

Il est aussi possible d'ordonner les éléments réponses suivant un seul score, noté $E$, qui est l'agrégation des degrés de satisfaction de la contrainte et du souhait. Dans [47, 46], diverses formules ont été proposées pour calculer ce score :

$$E(x) = \min(c(x), \lambda.c(x) + (1 - \lambda).w(x)) \tag{2.16}$$

$$E(x) = \min(c(x), \max(w(x), 1 - \min(c(x), \alpha))) \tag{2.17}$$

La formule (2.16) permet de favoriser les éléments satisfaisant les souhaits $w_i$ parmi ceux qui satisfont plus ou moins les contraintes $c_i$. La formule (2.17), quant à elle, permet d'effectuer une agrégation hiérarchique entre $c$ et $w$, où la contrainte $c$ doit être satisfaite avec une priorité égale à 1 et, si la contrainte est satisfaite alors le souhait $w$ doit l'être aussi avec une priorité $\alpha$ moindre.

Cependant, le calibrage des deux valeurs $\lambda$ et $\alpha$ est très délicat. De plus, les auteurs ont montré dans [47, 46] que ces agrégations souffrent de quelques inconvénients. En effet, l'agrégation des deux degrés $c(x)$ et $w(x)$ peut changer l'interprétation de la condition bipolaire « *c et si possible w* » en une conjonction « *c et w* » des deux conditions "contrainte $c$" et "souhait $w$". De plus, des classements incohérents peuvent être engendrés lorsqu'un même degré de satisfaction est obtenu pour deux couples de degrés complètement différents.

#### 2.3.2.2 Approche hiérarchique

Dans [97, 98, 130, 131], les auteurs s'intéressent à l'expression et à l'évaluation des conditions bipolaires floues pour l'interrogation flexible des bases de données. Ils traitent deux types de conditions bipolaires, à savoir, (i) les conditions bipolaires de nature conjonctive ayant la forme « *c et si possible w* » et (ii) les conditions bipolaires de nature disjonctive de la forme « *e ou à défaut f* ».

La première forme « *c et si possible w* », s'interprète « *satisfaire c et si possible satisfaire w* », notée $(c, w)$. La condition floue $c$ représente une condition obligatoire exprimant une contrainte que tous les éléments filtrés doivent respecter (i.e., les éléments acceptables) et la condition $w$ correspond à un souhait permettant de désigner parmi les éléments acceptables, ceux qui sont les plus souhaités (i.e., les éléments optimaux). Quant à la deuxième forme « *e ou à défaut f* », notée $[e, f]$, elle s'interprète « *satisfaire e, ou à défaut satisfaire f* » dans laquelle la condition floue « $e$ » exprime un souhait qui représente les éléments les plus désirés (i.e., les éléments optimaux) et la condition « $f$ » correspond à une contrainte déterminant les éléments acceptés, tel que tout élément ne satisfaisant pas cette condition est définitivement rejeté.

Ces deux formalismes de conditions bipolaires floues vérifient la propriété de cohérence, où $\forall x, \mu_w(x) \leq \mu_c(x)$ et $\forall x, \mu_e(x) \leq \mu_f(x)$. Toutefois, ils sont différents dans l'expression des ces deux types de conditions bipolaires. Dans le premier type « *c et si possible w* », la contrainte $c$ est considérée plus importante que le souhait $w$ et donc la satisfaction de la condition d'acceptation $c$ est privilégiée, tandis que la deuxième forme « *e ou à défaut f* » donne l'importance à l'optimalité et privilège la satisfaction de la condition $e$.

En étudiant les propriétés et l'interprétation de chacun de ces deux formalismes ainsi que leur compatibilité, ces deux derniers ont été généralisés dans un même modèle, noté $\langle a, b\rangle$, permettant d'exprimer d'une manière générale une condition bipolaire floue indépendamment de son formalisme (i.e., « *c et si possible w* » ou « *e ou à défaut f* », telle que la paire de degrés de satisfaction de $[0, 1]^2$ de chaque condition floue s'interprète comme suit :

- $\langle \alpha, \beta\rangle$ avec $\alpha < \beta$ représente la satisfaction d'une condition bipolaire de type « *ou à défaut* »
- $\langle \alpha, \beta\rangle$ avec $\alpha = \beta$ représente la satisfaction d'une condition floue
- $\langle \alpha, \beta\rangle$ avec $\alpha > \beta$ représente la satisfaction d'une condition bipolaire de type « *et si possible* ».

Ainsi, pour évaluer une requête combinant ces deux formalismes, les auteurs proposent deux opérateurs basés sur l'ordre lexicographique, appelés respectivement « *lmin* » et « *lmax* », permettant la définition de la conjonction et de la disjonction des deux conditions bipolaires. Ces deux opérateurs correspondent respectivement à une t-norme et t-conorme étendues.

L'opérateur *lmin* est une fonction définie par :

$$\begin{aligned} lmin : ([0,1]\times[0,1])^2 &\rightarrow [0,1]\times[0,1] \\ \left(\langle\alpha,\beta\rangle, \langle\alpha',\beta'\rangle\right) &\mapsto lmin\left(\langle\alpha,\beta\rangle, \langle\alpha',\beta'\rangle\right) = \begin{cases} \langle\alpha,\beta\rangle & \text{si } (\alpha<\alpha')\vee(\alpha=\alpha')\wedge\beta<\beta' \\ \langle\alpha',\beta'\rangle & \text{sinon} \end{cases} \end{aligned}$$

L'opérateur *lmax* correspond à une fonction de la forme :

$$\begin{aligned} lmax : ([0,1]\times[0,1])^2 &\rightarrow [0,1]\times[0,1] \\ \left(\langle\alpha,\beta\rangle, \langle\alpha',\beta'\rangle\right) &\mapsto lmax\left(\langle\alpha,\beta\rangle, \langle\alpha',\beta'\rangle\right) = \begin{cases} \langle\alpha,\beta\rangle & \text{si } (\alpha>\alpha')\vee(\alpha=\alpha')\wedge\beta>\beta' \\ \langle\alpha',\beta'\rangle & \text{sinon} \end{cases} \end{aligned}$$

En outre la négation de la condition bipolaire floue correspondante se traduit comme suit : $\neg \langle \alpha, \beta \rangle = \langle \neg\alpha, \neg\beta \rangle$. Ceci a permis de montrer que la négation d'une condition de type « et si possible » correspond à une condition bipolaire de type « ou à défaut », et réciproquement. Voir [97], pour plus de détails.

#### 2.3.2.3 Approche basée sur l'opérateur « winnow »

Dans le contexte de l'interrogation flexible des bases de données relationnelles, une approche basée sur l'opérateur « *winnow* », a été proposée dans [150, 151]. Cet opérateur est à l'origine introduit par Chomicki [30], pour sélectionner les meilleurs éléments répondant à une requête à préférences, en se basant sur une relation de préférence $R$. En appliquant cette dernière à un ensemble d'éléments $T$, l'opérateur « *winnow* », noté $w_R$, peut être défini comme suit :

$$w_R(T) = \{t \in T | \neg\exists t' \in T, t' \succ_R t\} \tag{2.18}$$

où $t \succ_R t'$ signifie que « $t$ *est préféré à* $t'$ » suivant la relation de préférence $R$.

En d'autres termes, la formule (2.18) permet de retourner tout élément de l'ensemble $T$ tel qu'il n'existe pas un autre élément $t' \in T$ qui lui soit préféré. L'opérateur « *winnow* » sélectionne donc les éléments de $T$ qui ne sont dominés par aucun autre élément suivant la relation $R$.

Cependant, dans [30], il n'existe aucune formulation distinguant les contraintes des souhaits. Par conséquent, Zadrożny et Kacprzyk [150, 151] ont étendu cet opérateur, dit « *winnow* », aux requêtes à préférences formulées cette fois-ci avec des conditions bipolaires. Les requêtes bipolaires traitées sont de la forme « *sélectionner les éléments vérifiant les contraintes "c" et si possible les souhaits "w"* ».

**1. Opérateur « winnow » pour des conditions booléennes**

Dans le cas où les contraintes "$c$" et les souhaits "$w$" sont des conditions booléennes, la requête bipolaire se traduit par l'opérateur suivant :

$$w_R(\sigma_c(T)) = \{t \in \sigma_c(T) | \neg\exists t' \in \sigma_c(T), w(t') \wedge \neg w(t)\} \tag{2.19}$$

où $\sigma_c$ est une sélection sur la contrainte "$c$" retournant l'ensemble $\sigma_c(T)$ des éléments $t \in T$ vérifiant cette dernière. On extrait parmi les éléments satisfaisant la contrainte "$c$" ceux qui vérifient le souhait "$w$", s'il y en a, sinon les éléments vérifiant seulement la contrainte sont retournés.

Cette opérateur peut se traduire par un opérateur « winnow » :

$$w_R(\sigma_c(T)) = \{t \in \sigma_c(T) | \neg\exists t' \in \sigma_c(T), t' \succ_R t\} \tag{2.20}$$

avec $R$ est une relation de préférence telle que $t \succ_R t'$ est équivalent à $w(t) \wedge \neg w(t')$ et, $w(t)$ et $w(t')$ signifient que $t$ et $t'$ satisfont le souhait $w$.

Autrement dit, cette extension permet d'abord de filtrer, à l'aide de l'opérateur de sélection $\sigma_c$, l'ensemble des éléments vérifiant la contrainte "$c$", ensuite, la relation de préférence $R$ est appliquée

sur cet ensemble pour extraire les éléments vérifiant le souhait $w$.

Cependant, il est important de noter que cette approche traitant, comme celle dans [47, 46], des requêtes bipolaires composées de conditions obligatoires, dites contraintes, et de conditions optionnelles, dites souhaits, n'impose aucune condition de cohérence entre les contraintes et les souhaits. De plus, elle considère que les souhaits sont plus prioritaires que les contraintes, car on rejette les éléments qui satisfont seulement la contrainte, lorsqu'il existe un autre élément qui satisfait à la fois la contrainte et le souhait. Toutefois, les résultats retournés par l'opérateur « *winnow* » exprimé dans la formule (2.19), peut ne pas correspondre aux attentes de l'utilisateur qui considère que le souhait n'est qu'une condition optionnelle et non pas obligatoire.

**2. Opérateur « *winnow* » pour des conditions floues**

Dans [150, 151, 152], les auteurs introduisent l'équivalent flou de l'opérateur précédent en se basant sur une relation de préférence floue $R$. Il permet donc de calculer un degré d'appartenance pour chaque élément non dominé à l'aide de la formule suivante :

$$\mu_{w_R(\sigma_c(T))} = truth(T(t) \wedge \forall t' \in T\,(T(t') \rightarrow \neg R(t,t'))) \tag{2.21}$$

où $R$ est une relation de préférence floue, $T(t)$ et $T(t')$ sont les degrés d'appartenance respectifs des éléments $t$ et $t'$ à l'ensemble des éléments $T$, $\wedge$ est une t-norme et $\rightarrow$ est une implication floue. Le résultat de la formule (2.21), noté $\mu_{w_R(\sigma_c(T))}$, représente le degré de vérité correspondant à l'appartenance d'un élément $t$ à l'ensemble flou des éléments définis par $w_R(T)$. Autrement dit, il exprime le degré de vérité de l'expression « $t$ satisfait le prédicat flou de $T$ et tous les autres éléments $t'$ de $T$ ne dominent pas $t$ ».

Dans le cas où les conditions d'une requête bipolaire de la forme « $c$ *et si possible* $w$ » sont exprimées à l'aide de prédicats flous, Zadrożny et Kacprzyk [150, 151, 152] proposent une extension de la formule (2.19) comme suivant :

$$w_R(\sigma_c(T)) = c(t) \wedge \forall t' \in T\,(c(t') \rightarrow \neg(w(t) \wedge \neg w(t'))) \tag{2.22}$$

où $R$ est une relation de préférence floue de la forme $R(t,t') \Leftrightarrow (w(t) \wedge \neg w(t'))$, $c(t)$ signifie que l'élément $t$ satisfait la contrainte $c$, $w_R(\sigma_c(T))$ représente l'ensemble des éléments non dominés de $C(T)$ suivant la relation floue $R$ tel que $C(T)$ représente l'ensemble flou des éléments de $T$ vérifiant la contrainte "$c$" et, $w(t)$ et $w(t')$ signifient que $t$ et $t'$ satisfont le souhait $w$.

La formule (2.22) peut être considérée comme étant une généralisation de la formule (2.19), car l'ensemble des éléments retournés par cette dernière sont identiques à ceux retournés par la formule (2.22) dans le cas où les conditions bipolaires sont booléennes. Néanmoins, nous constatons les mêmes inconvénients que ceux présentés pour un opérateur « *winnow* » appliqué sur des conditions booléennes.

D'autres approches bipolaires sont possibles, Tamani en a fait une étude comparative. Pour plus de détails sur ces méthodes, voir [131].

## 2.4 Approches non-agrégatives

Parmi les modèles d'évaluation de requêtes à préférences, on distingue des approches fondées sur le principe de Pareto, à savoir, le modèle Skyline [14] et le modèle PreferenceSQL [80]. Elles se basent sur des relations de préférences binaires non commensurables pour retourner les éléments qui ne sont dominés par aucun autre élément du même ensemble de données.

Dans la sous-section 2.4.1, nous présentons le modèle Skyline ainsi que les différentes approches proposées pour soit raffiner, soit relaxer les requêtes skylines. Quant au modèle PreferenceSQL, il est résumé en sous-section 2.4.2.

### 2.4.1 Modèle Skyline seulement

Dans l'objectif de rendre efficace les applications de prise de décisions multicritères, Borzsonyi *et al.* [14] étendent les systèmes d'interrogation de bases de données par un opérateur, dit ***Skyline***. Ce dernier permet de filtrer un ensemble de réponses, considérées intéressantes du point de vue de l'utilisateur, à partir d'un ensemble de données potentiellement important.

Pour comparer deux tuples de la base suivant une requête à préférences, les valeurs de chaque tuple sur chaque dimension (critère, attribut) sont comparées deux à deux. Par conséquent, un ordre partiel peut être défini sur ces tuples, en appliquant l'ordre de ***Pareto***.

#### 2.4.1.1 Skyline au sens de Pareto

Les requêtes skylines [14] sont un exemple spécifique et bien pertinent des requêtes à préférences. Elles permettent de sélectionner l'ensemble des points considérés les plus intéressants, lorsque différents critères et souvent conflictuels sont pris en compte. Elles s'appuient sur le principe de dominance au sens de Pareto qui peut être défini comme suit :

**Définition 2.3.1 (Principe de dominance au sens de Pareto)**. Soit $D$ un ensemble de points $d$-dimensionnels et, $p_i$ et $p_j$ deux points de $D$. On dit que $p_i$ **domine** (**au sens de Pareto**) $p_j$ si et seulement si $p_i$ est meilleur ou égal à $p_j$ sur toutes les dimensions et strictement meilleur que $p_j$ sur au moins une dimension.

On dit alors que $p_i$ *domine* (*est préféré à*) $p_j$ et on note $p_i \succ p_j$.

**Définition 2.3.2 (Skyline)**. Le skyline $S$ de $D$ est l'ensemble des points, dits *points skyline*, qui ne sont dominés par aucun autre point de $D$ :

$$S = \{p \in D | \nexists p' \in D, p' \succ p\}$$

Les requêtes skyline calculent donc l'ensemble des tuples (points, éléments, objets) optimaux au sens de Pareto dans une relation, i.e., les tuples qui ne sont dominés par aucun autre tuple de la

même relation. De plus, il a été montré dans [30] que l'opérateur Skyline est un cas particulier de l'opérateur winnow.

**Exemple 2.3.1** Soit une base de données contenant des informations sur des hôtels comme indiqué dans le tableau 2.1 (où la dimension $d = 2$).

TABLE 2.1 – Exemple d'hôtels.

| Hôtel | Prix (€) | Distance (Km) |
|---|---|---|
| $p_1$ | 60,00 | 150 |
| $p_2$ | 50,00 | 110 |
| $p_3$ | 45,00 | 240 |
| $p_4$ | 40,00 | 180 |
| $p_5$ | 37,00 | 270 |
| $p_6$ | 30,00 | 195 |
| $p_7$ | 32,00 | 210 |

Considérons une personne qui cherche un hôtel aussi proche que possible de la plage et ayant un prix faible. On peut vérifier que le skyline résultant $S$ contient les hôtels (points, tuples) : $p_2$, $p_4$ et $p_6$. Par exemple, $p_1$ est dominé par $p_2$ et, $p_7$ est dominé par $p_6$. Le tuple $p_2$ appartient au skyline $S$ parce qu'il est l'élément le plus intéressant suivant le critère "Distance", $p_6$ appartient au skyline $S$ car il est le tuple le plus satisfaisant suivant le critère "Prix" et enfin, $p_4$ apparait dans le skyline $S$ parce qu'il représente un bon compromis entre les deux critères "Prix" et "Distance". □

#### 2.4.1.2 Raffinement et Relaxation de Skyline

Plusieurs études ont été menées pour développer des algorithmes efficaces et définir des variantes pour les requêtes skyline [111, 147, 79, 69]. Toutefois, l'interrogation d'un ensemble de données multidimensionnelles à l'aide de l'opérateur Skyline, peut conduire à deux scénarios : soit (i) un nombre important de réponses est retourné, ce qui est généralement peu informatif du point de vue de l'utilisateur, ou bien (ii) un nombre insuffisant de réponses est sélectionné, ce qui peut être aussi insuffisant du point de vue de l'utilisateur.

Pour remédier à ces deux problèmes, différentes méthodes ont été introduites pour soit raffiner le skyline, donc réduire sa taille (cas (i)) [53, 25, 24, 99, 73, 50, 68, 110], soit relaxer le skyline pour retourner plus de réponses à l'utilisateur (cas (ii)) [62, 68].

Dans ce qui suit, nous présentons un état de l'art des diverses méthodes proposées dans la littérature, pour le raffinement et la relaxation du skyline. Nous commençons par les approches de raffinement, dans le cas où le skyline est de taille assez importante, et par la suite, les modèles permettant d'augmenter la taille du skyline dans le cas où elle est insuffisante, sont introduites.

– **Raffinement de Skyline**

Plus le nombre de dimensions augmente, plus le nombre de points retournés par le skyline devient très important et peut être moins informatif. Du point de vue de l'utilisateur, il est donc important

de raffiner et de réduire la taille du skyline en lui sélectionnant les points les plus intéressants.

Dans ce contexte, de nombreuses approches ont été proposées pour réduire la taille du skyline [110, 24, 99, 73, 68]. Elles proposent des métriques et des critères pour identifier les points les plus satisfaisants, en sélectionnant les top-k réponses. Cependant, les critères d'ordonnancement considérés peuvent ne pas correspondre aux besoins de l'utilisateur.

**Approche des k-dominants.** Papadias *et al.* [110] proposent la notion des *k-dominants*, qui détermine un ensemble $T$ de $k$ points tels que la somme des cardinalités des ensembles des objets dominés par chaque élément de $T$ (i.e. $\sum_{p \in T} |Dom(\{p\})|$ où $Dom(\{p\})$ représente l'ensemble des objets dominés par l'objet $p$), est maximisée. Cependant, cette méthode ne prend pas en considération le cas où un objet peut être dominé par plusieurs autres objets et peut retourner des objets qui ne sont pas des points du skyline.

**Approche des top-k représentatifs points skyline**. Dans [99], les auteurs déterminent un ensemble $L$ contenant $k$ points du skyline, dits *top-k représentatifs points du skyline*, de telle sorte que la somme des cardinalités des ensembles des objets dominés par chaque point skyline de $L$, est maximisée. En d'autres termes, la somme $\sum_{p \in L} |Dom(\{p\})|$ où $Dom(\{p\})$ représente l'ensemble des objets dominés par l'objet skyline $p$, doit être maximale.

Considérons la base de données 3-dimensionnelles $D = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$ présentée dans la table 2.2. Par souci de simplicité et sans perte de généralité, nous supposons dans cette sous-section que plus la valeur $p_i$ est grande sur chaque dimension, meilleure elle est. Ainsi, conformément aux définitions 2.3.1 et 2.3.2, le skyline de l'ensemble $D$ est composé des trois points $p_1(0.8, 0.3, 0.7)$, $p_2(0.2, 0.6, 0.8)$ et $p_3(0.9, 0.9, 0.5)$.

TABLE 2.2 – Exemple de bases de données $D$.

| | $d_1$ | $d_2$ | $d_3$ |
|---|---|---|---|
| $p_1$ | 0.8 | 0.3 | 0.7 |
| $p_2$ | 0.2 | 0.6 | 0.8 |
| $p_3$ | 0.9 | 0.9 | 0.5 |
| $p_4$ | 0.8 | 0.2 | 0.4 |
| $p_5$ | 0.6 | 0.3 | 0.6 |
| $p_6$ | 0.7 | 0.3 | 0.5 |
| $p_7$ | 0.2 | 0.5 | 0.7 |
| $p_8$ | 0.1 | 0.6 | 0.6 |

Par application de l'approche proposée dans [99], l'ensemble des top-k(=2) représentatifs points skyline de $S$ est représenté par l'ensemble $L = \{p_1, p_2\}$. En effet, la somme des cardinalités des ensembles des objets dominés par chaque point de $L$, est maximisée et, égale à $|\{p_4, p_5, p_6, p_7, p_8\}| = 5$, car $|Dom(\{p_1\})| = |\{p_4, p_5, p_6\}| = 3$, $|Dom(\{p_2\})| = |\{p_7, p_8\}| = 2$ et $|Dom(\{p_3\})| = |\{p_4, p_6\}| = 2$.

Toutefois, en comparant les valeurs de l'objet $p_2$ avec celle de $p_3$, nous constatons que l'élément $p_3$ est beaucoup plus intéressant que $p_2$. De plus, cette approche [99] est NP-complet dans le cas d'au moins trois dimensions.

**Approche des top-k points fréquents.** Afin de raffiner un skyline de taille importante, Chan *et al.* [25] proposent une métrique, dite *fréquence du skyline*, « *skyline frequency* » en anglais, pour calculer les top-k objets fréquents du skyline. Pour cela, ils mesurent, pour chaque objet $p$, le nombre des sous-espaces de dimensions où $p$ est un point du skyline, pour choisir enfin les k points les plus fréquents.

L'inconvénient majeur de cette métrique réside dans le fait que certains des points les plus intéressants peuvent être classés au même niveau que d'autres points beaucoup moins intéressants par rapport à la requête de l'utilisateur. De plus, en pratique, la méthode proposée est très coûteuse car elle calcule les skylines de $2^d - 1$ sous-ensembles d'un espace $d$-dimensionnel.

**Approche des k-dominants points skyline.** Dans [24], Chan *et al.* proposent le critère, dit *la k-dominance*, qui calcule l'ensemble des points (du skyline) qui ne sont dominés par aucun autre point dans tous les sous-espaces de $k$ dimensions. Un point $p$ *k-domine* un autre point $p'$, s'il existe $k$ ($\leq d$) dimensions, sur lesquelles $p$ est aussi bien que $p'$ et strictement meilleur sur *au moins une de ces k* dimensions. Ainsi, un point $p_i$ appartient au skyline des k-dominants si et seulement s'il n'existe aucun autre point $p_j$ (tel que $i \neq j$) dans la base de données qui le $k$-domine.

Si nous devons raffiner, par exemple, un skyline composé des deux points $p_1(0.5, 0.7, 0.4, 0.3, 0.8)$ et $p_2(0.4, 0.6, 0.4, 0.5, 0.6)$ en appliquant la relation k-dominance avec $k = 2$, nous éliminerons les deux points, car $p_1$ k(=2)-domine $p_2$ et $p_2$ k(=2)-domine $p_1$. Cependant, la comparaison des valeurs des deux points fait constater que $p_1$ est plus intéressant que $p_2$, car il est strictement meilleur que $p_2$ sur trois dimensions parmi cinq, tandis que $p_2$ n'est meilleur que $p_1$, que sur une seule dimension. En d'autres termes, au lieu de retourner un ensemble vide de réponses, le point skyline $p_1$ devrait être retourné comme résultat car il répond mieux à la requête de l'utilisateur sur plus de dimensions.

**Approche basée sur le critère de diversité.** Dans [73], le concept de skyline de similarité a été proposé et son application est dédiée aux bases de cas. Il permet de retourner le sous-ensemble de cas maximalement similaires à une requête donnée, en utilisant la dominance au sens de Pareto.

Pour réduire le nombre de résultats retournés par le skyline, les auteurs appliquent le critère de diversité, pour sélectionner un sous-ensemble (de taille "k") de cas qui est aussi divers que possible. Toutefois, les réponses retournées peuvent être celles qui satisfont le moins la requête de l'utilisateur sur la plupart des dimensions (comme celles qui sont meilleures juste sur une seule dimension). Enfin, cette approche nécessite le calcul de la diversité de tous les sous-ensembles de taille "k" possibles, ce qui est une tâche très coûteuse.

**Approche fondée sur la notion de skylines flous.** Dans [68], les auteurs introduisent la notion de *skyline graduel*. Ce dernier est le résultat d'un raffinement du skyline original (i.e. obtenu à l'aide de la relation de dominance au sens de Pareto), dans lequel l'appartenance d'un élément au skyline raffiné est exprimée à l'aide d'un degré.

L'élément clé de cette approche est une relation floue qui permet d'exprimer dans quelle mesure un élément est « *beaucoup plus préféré à* » un autre élément. Un point $p$ appartient donc au skyline graduel, noté $S_g$, si $p$ est un point skyline et $\forall p' \in Dom(p)$ (où $Dom(\{p\})$ représente l'ensemble des objets dominés par l'objet $p$), il existe au moins une dimension $d_i$ telle que la valeur de $p$ sur cette dimension est beaucoup plus préféré à celle de $p'$ sur la même dimension, c'est-à-dire : $p \in S_g \Leftrightarrow p \in S \wedge \forall p'(p'_1, ..., p'_d) \in Dom(p), \exists i \in \{1, ..., d\}$ tel que $p.d_i$ est beaucoup plus préféré à $p'.d_i$, où $p.d_i$ et $p'.d_i$ dénotent, respectivement, la valeur de $p$ et $p'$ sur la dimension $d_i$.

Néanmoins, le critère de raffinement appliqué n'est pas fondé sur la comparaison entre les points skyline, mais plutôt sur la comparaison entre chaque point skyline $p_i \in S$ et les points $p_j$ qu'il domine (i.e. $p_j \in Dom(p_i)$), pour évaluer dans quelle mesure $p_i$ est bien préféré à chaque point $p_j$.

Considérons, par exemple, les deux points skyline $p_1(0.9, 0.9, 0.5)$ et $p_2(0.7, 0.5, 0.6)$ de la base de données $D' = \{p_1, p_2, p_3, p_4, p_5\}$ illustrée dans le tableau 2.3, où l'ensemble $Dom(p_1)$ des points dominés par $p_1$ est composé de $p_3$ et, celui de $p_2$ contient $p_4$ et $p_5$.

TABLE 2.3 – Exemple de bases de données $D'$.

| | $d_1$ | $d_2$ | $d_3$ |
|---|---|---|---|
| $p_1$ | 0.9 | 0.9 | 0.5 |
| $p_2$ | 0.7 | 0.5 | 0.6 |
| $p_3$ | 0.8 | 0.8 | 0.4 |
| $p_4$ | 0.6 | 0.2 | 0.6 |
| $p_5$ | 0.4 | 0.5 | 0.6 |

Supposons maintenant que la relation « beaucoup plus préféré à » est définie comme suit : $p$ est beaucoup plus préféré à $p'$ si est seulement si $\exists i \in \{1, ..., 3\}$ tel que $p.d_i - p'.d_i \geq 0.3$. Par conséquent, l'application de la méthode proposée dans [68] construit un skyline raffiné composé du point $p_2$. Cependant, en comparant les valeurs de $p_2$ avec celles de $p_1$, nous constatons que $p_1$ est plus intéressant que $p_2$, car $p_1$ est bien meilleur sur plus de dimensions.

– **Relaxation de Skyline**

Dans le domaine des requêtes skyline, la relaxation est un processus important et nécessaire dans le cas où le nombre de réponses retournées est insuffisant du point de vue de l'utilisateur. À titre d'exemple, un recruteur souhaitant embaucher cinq ingénieurs en informatique, peut ne pas être satisfait d'un résultat inférieur à cinq candidats. Pour remédier à ce problème, une solution est de rendre plus flexible la relation de dominance au sens de Pareto et d'enrichir le résultat obtenu avec les réponses les plus intéressantes par rapport à la requête de l'utilisateur.

Dans ce contexte, peu de travaux ont été réalisés pour retourner plus de réponses à l'utilisateur

que celles sélectionnées par le skyline. Goncalves et Tineo [62] proposent une relation de dominance flexible en utilisant des opérateurs flous de comparaison. Cette approche permet donc d'augmenter le skyline avec des points faiblement dominés par tout autre point. Dans [68], quelques propositions de relaxation ont été aussi évoquées.

### 2.4.2 Modèle PreferenceSQL

Dans le contexte d'interrogation de bases de données relationnelles, Kießling et Köstler [81] ont introduit un langage formel, appelé *PreferenceSQL*, fondé sur un ensemble d'opérateurs permettant aux utilisateurs d'exprimer leurs préférences. Ce modèle qui est une extension du langage SQL en intégrant le modèle algébrique présenté dans [80], permet donc de construire des préférences atomiques (simples) ou complexes, au moyen de différents constructeurs de préférences et les ordonner par ordre partiel strict suivant leurs distances (par rapport à des valeurs optimales) ou bien leurs degrés de satisfaction (par rapport à leur niveau de satisfaction).

En PreferenceSQL, une requête utilisateur peut avoir deux types de préférences : (i) les préférences soft (optionnelles) et (ii) les préférences obligatoires :

| | |
|---|---|
| **SELECT** | < informations à sélectionner > |
| **FROM** | < listes des relations interrogées > |
| **WHERE** | < préférences obligatoires> |
| **PREFERRING** | < préférences soft > |

La clause WHERE permet donc de sélectionner les éléments satisfaisant les préférences obligatoires qui sont des conditions booléennes, et la clause PREFERRING permet d'exprimer des préférences optionnelles permettant de discriminer entre les éléments filtrés par la clause précédente.

**Définition 2.3.3 (Préférence Obligatoire).** Une préférence obligatoire (*constraint* ou *hard preference*, en anglais) est une expression relationnelle de la forme *(m, o, r)*, où $o \in \{<, >, \leq, \geq, =, \neq\}$ est un opérateur relationnel et $r$ est une valeur d'un élément sur l'attribut $m$. Ce type de préférence permet de filtrer les éléments répondant à la contrainte de l'utilisateur.

**Définition 2.3.4 (Préférence soft).** Une préférence soft est une expression pouvant avoir l'une des formes suivantes [80] :

- Préférences atomiques :
  - *Around(m, $r_{souhaité}$) :* pour un attribut $m$, cette expression favorise la valeur $r_{souhaité}$, sinon la valeur la plus proche de $r_{souhaité}$ est acceptée.
  - *Between(m, $r_{low}$, $r_{up}$) :* pour l'attribut $m$, cette expression favorise les valeurs dans l'intervalle $[r_{low}, r_{up}]$, sinon elle favorise les valeurs proches des bornes.
  - *Highest(m) :* pour un attribut $m$, cette expression favorise la valeur la plus élevée, sinon, la valeur la plus proche du maximum est favorisée. Par exemple, le maximum de l'attribut « disponibilité d'un service Web» est égal à 100%.

- *Lowest(m) :* pour un attribut $m$, cette expression favorise la valeur la plus petite, sinon, la valeur la plus proche du minimum est favorisée, par exemple : le minimum de l'attribut « prix » est égal à 0.
- *Pos(m, $S_1$)* : pour un attribut $m$, cette expression favorise une des valeurs de l'ensemble $S_1$, sinon n'importe quelle autre valeur est acceptée.
- *Neg(m, $S_2$) :* pour un attribut $m$, cette expression favorise les valeurs différentes de celles figurant dans l'ensemble $S_2$, sinon l'une des valeurs de $S_2$ est acceptée.

– Préférences complexes :

- *Pareto($p_i$, $p_j$) :* cette expression signifie que les deux préférences soft $p_i$ et $p_j$ ont la même importance.
- *Prioritized($p_i$, $p_j$) :* cette expression signifie que la préférence soft $p_i$ est plus importante que la préférence soft $p_j$.

Afin d'évaluer un élément "$x$" donné suivant les préférences d'un utilisateur, le modèle PreferenceSQL calcule la distance entre chaque préférence atomique "$p$" et la valeur, notée $r$, de cet élément sur l'attribut (critère, dimension) "$m$" correspondant. Le tableau 2.4 résume quelques fonctions de calcul de distance utilisées dans [81] pour évaluer une requête à préférence donnée.

TABLE 2.4 – Quelques fonctions de calcul de distances selon le modèle PreferenceSQL.

| Préférence $p$ sur $m$ | Distance de satisfaction de $p$ sur $m$ |
|---|---|
| Around$(m, r_{souhaité})$ | $d(p,x) = \lvert r - r_{souhaité} \rvert$ |
| Highest$(m)$ | $d(p,x) = r_{max}–r$, où $r_{max}$ est la plus grande valeur |
| Lowest$(m)$ | $d(p,x) = r–r_{min}$, où $r_{min}$ est la plus petite valeur |
| Between$(m, r_{low}, r_{up})$ | $d(p,x) = \begin{cases} 0, & r \in [r_{low}, r_{up}] \\ r_{low}–r, & r < r_{low} \\ r - r_{up}, & r > r_{up} \end{cases}$ |

À partir du tableau ci-dessus, nous constatons que la mesure $d(p,x)$, exprimant la distance de satisfaction entre la préférence $p$ et l'élément "$x$" sur l'attribut $m$, dépend du type de constructeur de préférences (i.e., around, between, ...) utilisé pour définir la préférence $p$.

Cependant, une stratégie de combinaison et d'agrégation entre les différentes conditions atomiques composant une requête utilisateur, n'est pas possible, car PreferenceSQL utilise des fonctions différentes pour évaluer dans quelle mesure un tuple satisfait une condition atomique. En d'autres termes, les valeurs calculées pour évaluer à quel point un élément satisfait les préférences d'une requête donnée, ne sont pas commensurables. Par ailleurs, les meilleurs éléments au sens de Pareto sont retournés à l'utilisateur sans savoir à quel point un élément est meilleur qu'un autre élément.

## 2.5 Conclusion

Dans ce chapitre, nous avons présenté quelques approches portant sur l'interrogation *à préférences* des bases de données relationnelles. Ces approches sont utilisées dans la suite de notre travail et

permettent l'expression et l'évaluation des préférences utilisateur, spécifiées sur différents critères de sélection. Cette étude nous a permis de voir l'importance des préférences dans l'interrogation des bases de données pour, d'une part, retourner les meilleurs éléments du point de vue de l'utilisateur et, d'autre part, éviter les réponses vides.

Deux types d'approches d'évaluation de requêtes à préférences ont été abordées : (i) les approches agrégatives basées sur la théorie des ensembles flous et, (ii) les approches non-agrégatives fondées sur le principe de Pareto. Les modèles basées sur la théorie des ensembles flous permettent d'attribuer un seul degré à chaque élément évalué de la base, pour ensuite ordonner tous les éléments du plus satisfaisant au moins satisfaisant. Ces modèles ont la capacité d'évaluer non seulement des préférences à conditions booléennes mais aussi des préférences exprimées au moyen de prédicats flous et/ou de quantificateurs linguistiques. Quant aux approches fondées sur le principe de Pareto, elles déterminent l'ensemble des éléments non-dominés de la base, sans savoir réellement à quel point un élément est meilleur qu'un autre élément.

Nous remarquons, suite à cette étude de l'état de l'art, que la prise en considération des préférences dans les requêtes des utilisateurs joue un rôle important dans le domaine des bases de données relationnelles pour améliorer la qualité des résultats. Dans le domaine des bases de données de graphes, la plupart des approches ont eu recours à des techniques permettant d'améliorer la performance et la pertinence des appariements. Cependant, nous remarquons qu'il n'existe actuellement aucun outil d'appariement capable de prendre en considération à la fois, les correspondances entres les graphes et les préférences de l'utilisateur pour améliorer la qualité des résultats.

Pour contribuer à résoudre ces problèmes, nous proposons dans les parties suivantes, la prise en considération des préférences utilisateur dans le cadre des bases de données autre que relationnelles, à savoir les bases de données d'objets complexes. Pour cela, nous proposons, dans ce qui suit, des approches d'évaluation de requêtes à préférences dans les deux cas : l'agrégation et la non-agrégation des préférences.

# Deuxième partie

# Contributions

# Chapitre 3

# Découverte et sélection de modèles de processus en présence de préférences

## 3.1 Introduction

Comme discuté dans le chapitre 1, la structure de données « *graphe* » est utilisée dans différents domaines d'application pour représenter le contenu et la structure des objets complexes, comme les documents XML [89, 155], les processus métiers [58, 59], les composantes chimiques [82, 139], ... Dans le cadre de cette représentation, pour répondre à une requête utilisateur, modélisée sous forme de graphe, des techniques d'appariement de graphes sont alors requises.

Puisque le problème d'appariement de graphes est NP-complet, plusieurs travaux ont été menés pour (i) spécifier des algorithmes manipulant des graphes de taille raisonnable et (ii) appliquer des méthodes de classement des résultats issus du processus d'appariement. Dans le cadre de la découverte de services (modèles de processus) axée sur la qualité, domaine d'application de graphes, de nombreuses approches ont été proposées dans la littérature. Ces dernières diffèrent les unes des autres par la sémantique capturée par les graphes considérés dans le processus d'appariement et par les métriques de classement des résultats obtenus. La plupart d'entre elles sont fondées sur l'appariement des entrées/sorties du service [84], sur le comportement du service (décrit comme un processus métier) [41, 42, 67], ou encore sur une connaissance ontologique [67]. Toutefois, ces méthodes se caractérisent par un niveau de sélectivité très bas dû au grand nombre de services retournés et offrant des fonctionnalités et des comportements similaires.

Une façon de discriminer les services similaires est de considérer les caractéristiques non-fonctionnelles telles que les préférences sur les aspects de qualité de services (temps d'exécution, disponibilité, sécurité, etc.). Ainsi, pour une requête donnée dans un contexte donné, il n'est pas nécessaire de fournir tous les services de même fonctionnalité, mais plutôt seulement ceux qui satisfont le mieux les préférences et les contraintes contextuelles de l'utilisateur. Dans cette direction, un certain nombre d'approches ont vu le jour [105, 36, 158, 86] en se limitant qu'aux services atomiques (elles traitent chaque service comme étant une boite noire).

Pour outrepasser cette limitation aux services atomiques, une approche intégrant à la fois les

aspects fonctionnels et non-fonctionnels dans le processus d'appariement de graphes, est alors nécessaire dans le domaine de recherche de services. Pour cela, les défis suivants doivent être relevés :

- spécification des contraintes non-fonctionnelles à différents niveaux de granularité de la description fonctionnelle du service (i.e., sur le service lui même et sur les activités le composant) ;
- détermination d'un mécanisme permettant de définir des requêtes contenant des contraintes sur l'aspect fonctionnel et sur la qualité de services ;
- spécification de métriques pour mesurer le degré avec lequel un service satisfait les contraintes de qualité d'une requête utilisateur ;
- définition d'une méthodologie d'évaluation de services pour calculer d'une façon efficace *leur similarité en tenant compte des contraintes fonctionnelles et non-fonctionnelles* de l'utilisateur.

D'autres défis plus spécifiques liés aux contraintes non-fonctionnelles devraient être également pris en compte :

- les services sont déployés dans des environnements dynamiques et hétérogènes de telle sorte que leurs propriétés non-fonctionnelles sont souvent données ou dérivées avec des précisions différentes ;
- les utilisateurs ne sont pas toujours en mesure de spécifier précisément leurs contraintes non-fonctionnelles ;
- les utilisateurs ont des points de vue différents sur ce qui est un service satisfaisant selon le même ensemble de contraintes non-fonctionnelles ;
- le processus de découverte de services devrait éviter les réponses vides ou pléthoriques en raison de l'imprécision de la requête de l'utilisateur.

Les préférences sont un moyen naturel pour faciliter la spécification des contraintes non-fonctionnelles dans la requête de l'utilisateur. Elles peuvent être *flexibles*, d'une part, pour éviter des réponses vides dues aux contraintes très spécifiques de l'utilisateur, et d'autre part, pour fournir un ensemble adéquat de résultats pertinents, même lorsque l'utilisateur définit des contraintes assez générales.

Dans ce chapitre, nous présentons un cadre de recherche de services [4, 5, 6, 9, 7], réalisé au sein du Projet AOC [1] et en collaboration avec l'équipe PRiSM, partenaire aussi du projet. Ce cadre permet d'intégrer les préférences des utilisateurs dans la spécification des requêtes. L'objectif est d'améliorer la qualité des résultats retournés, en prenant en considération (i) les contraintes fonctionnelles, c'est-à-dire, la similarité structurelle entre deux modèles de processus et, (ii) les contraintes non-fonctionnelles, c'est-à-dire, la satisfaction des préférences sur les aspects de qualité de services. Pour cela, nous introduisons, dans un premier temps, un modèle formel pour représenter à l'aide des graphes (i) les modèles de processus avec leurs caractéristiques de qualité et (ii) les requêtes des utilisateurs en présence des préférences sur les aspects de qualité de services. Ensuite, nous décrivons une méthodologie d'évaluation permettant de calculer la similarité entre des modèles de processus cibles et une requête utilisateur, en tenant compte de leur similarité structurelle et également de la satisfaction des préférences liées à la qualité.

1. Projet ANR « Appariement d'Objets Complexes ».

Dans ce qui suit, la section 3.2 décrit le modèle défini par l'équipe PRiSM [90] permettant de représenter les modèles de processus. Dans les sections 3.3 et 3.4, nous présentons, respectivement, la manière dont les informations de qualité y sont spécifiées dans la description fonctionnelle des modèles de processus cibles et, la manière dont nous avons introduit les préférences des utilisateurs dans les requêtes. La section 3.5 discute la modélisation et l'évaluation des préférences utilisateur. Nous présentons également, en section 3.5, les mesures proposées pour l'évaluation des modèles de processus en prenant en considération à la fois les exigences fonctionnelles et les exigences non-fonctionnelles de l'utilisateur. La section 3.6 décrit la méthodologie d'évaluation des modèles de processus de l'approche proposée qui prend en compte la satisfaction des préférences sur la Qualité de Services (QdS) et la similarité structurelle. Enfin, la section 3.7 conclut le chapitre.

## 3.2 Représentation des modèles de processus

Les modèles de processus (services), appelés aussi graphes de processus, représentent des abstractions fondamentales permettant aux concepteurs d'applications de bien repérer et de capturer les caractéristiques dynamiques de leurs applications. La modélisation de processus est donc une tâche importante dans le domaine de l'ingénierie logicielle.

Durant ces dernières années, le nombre de modèles de processus a connu une grande croissance, car de plus en plus les entreprises investissent dans la gestion de leurs processus. Ces modèles (encore appelés référentiels) doivent permettre, par exemple, de rechercher un processus contenant un fragment donné dans un dépôt de modèles de processus (services). La recherche de modèles dans de tels dépôts est un enjeu pour le bon fonctionnement des architectures orientées modèles en particulier et du domaine des services en informatique en général. Ce problème a reçu récemment une attention particulière dans la communauté du « Service Computing » et ainsi de nombreuses approches ont été proposées.

Les premières approches de recherche de modèles de processus se sont basées sur les mots clés ou le profil des processus, c'est-à-dire, leur nom, leurs entrées et leurs sorties [84]. Néanmoins, cette solution s'est avérée inefficace car elle peut fournir des modèles de processus de fonctionnalités complètement différentes, comme elle peut aussi retourner un grand nombre de résultats de telle sorte qu'une analyse manuelle par l'utilisateur est fastidieuse.

Pour cela, d'autres méthodes ont vu le jour en proposant aux utilisateurs de fournir plus d'informations sur le processus recherché telles que son flux de données, son flux de contrôle, etc. Ainsi, la requête peut être modélisée sous forme de graphe représentant le modèle de processus souhaité par l'utilisateur. En conséquence, pour rechercher les modèles de processus cibles répondant à cette requête, un processus d'appariement de graphes est nécessaire.

En effet, il existe actuellement plusieurs langages, comme OWL-S [2] [101] et BPEL4WS [3] [11],

2. OWL-S, « Ontology Web Language for Services », est une ontologie de services qui offre un ensemble de constructions de langage de balisage pour décrire les propriétés et les capacités des services Web en forme interprétable sans ambiguïté par ordinateur.

3. BPEL4WS « Business Process Execution Language for Web Services ».

permettant de décrire les modèles de processus. Ces langages représentent chaque modèle par un ensemble d'activités atomiques combinées à l'aide de structures de flux de contrôle, où chaque activité est définie par son nom, ses entrées et ses sorties. Par conséquent, ces modèles peuvent être modélisés par un graphe (orienté) composé d'un ensemble de sommets représentant les activités et les nœuds connecteurs reliés par des arêtes. Ainsi, les nœuds activités symbolisent des opérations atomiques exécutées par le modèle de processus (e.g., réservation d'hôtel, paiement, ...), les nœuds connecteurs représentent le flux de contrôle (e.g., AND-Split, XOR-Split, Start, ...) et les arêtes déterminent le flux d'exécution des opérations.

Il est important de noter qu'il existe plusieurs types de nœuds connecteurs. Chaque graphe d'un modèle de processus possède un seul nœud connecteur de type « Start » et un seul de type « End », qui représentent, respectivement, le commencement et la terminaison de l'exécution du modèle de processus les contenant. Il peut avoir également des nœuds de type « XOR-Split » permettant d'effectuer un seul choix parmi une ou plusieurs branches. Ces dernières sont ensuite fusionnées par le nœud correspondant, qui est « XOR-Join ». Enfin, un graphe d'un modèle de processus peut aussi contenir des nœuds de type « AND-Split » permettant de déclencher l'exécution de toutes leurs branches sortantes qui seront fusionnées par la suite par un nœud de type « AND-Join ». Pour plus de détails sur les composants d'un modèle de processus, voir [58, 59].

**Exemple 3.2.1.** Le graphe illustré dans la figure 3.1 représente un modèle de processus permettant soit d'afficher les informations concernant les hôtels d'une ville donnée, soit d'effectuer une réservation dans un hôtel. Dans ce graphe, les nœuds activités sont représentés par des rectangles de couleur bleue avec leur nom, leurs entrées et leurs sorties et, les nœuds connecteurs sont modélisés par des cercles en couleur noire. □

Comme le montre l'exemple de la figure 3.1, le graphe d'un modèle de processus peut être composé d'un ou plusieurs blocs. Ces blocs peuvent être des séquences, des boucles ou des branches parallèles ou alternatives [92] :

- Une séquence représente une exécution séquentielle d'au moins deux blocs, où le nœud d'entrée du premier bloc et le nœud de sortie du dernier bloc sont, respectivement, les nœuds d'entrée et de sortie du bloc séquentiel.
- Un bloc boucle, comme son nom l'indique, est une boucle autour d'un bloc donné, où les nœuds XOR-Join et XOR-Split définissant la boucle sont, respectivement, les nœuds d'entrée et de sortie du bloc boucle.
- Un bloc parallèle, quant à lui, définit une exécution parallèle de deux ou plusieurs blocs avec les nœuds AND-Split et AND-Join qui représentent, respectivement, les nœuds d'entrée et de sortie du bloc parallèle. Le AND-Split déclenche les branches parallèles et le AND-Join les fusionne.
- Un bloc alternatif est une exécution alternative de blocs. Les nœuds d'entrée et de sortie de ce bloc alternatif sont, respectivement, le nœud XOR-Split déclenchant les branches alternatives et le nœud XOR-Join fusionnant ensuite ces dernières.

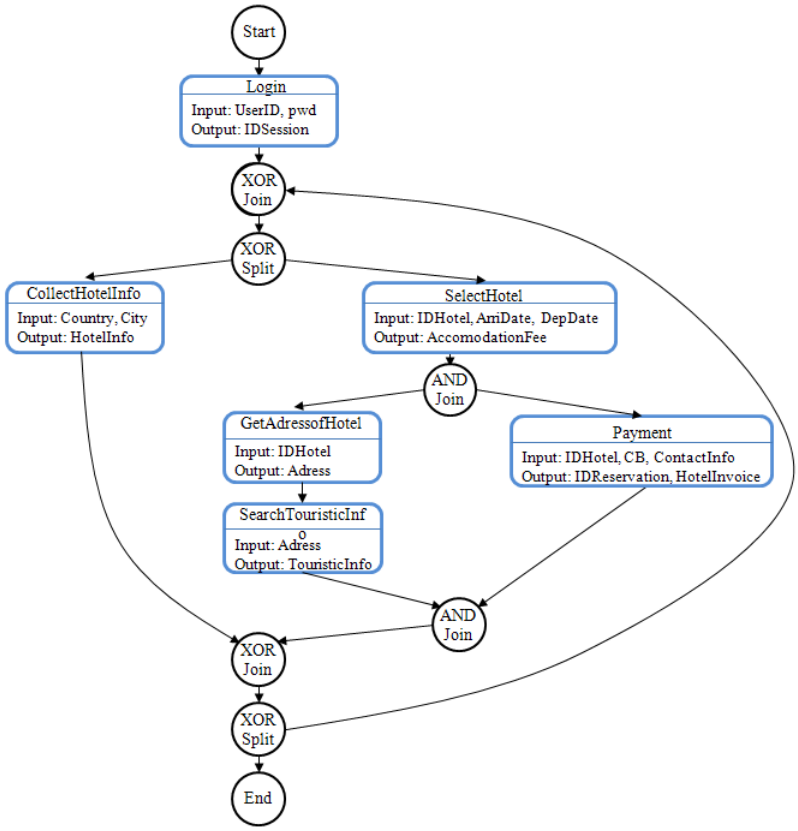



FIGURE 3.1 – Modèle de processus « *ReservationHotel* ».

**Exemple 3.2.2.** Dans le modèle de processus présenté dans la figure 3.1, nous remarquons, par exemple : un bloc séquentiel formé des deux activités « *getAdressofHotel* » et « *SearchTouristicInf* », un bloc alternatif défini à l'aide du premier nœud connecteur XOR-Split en commençant par le nœud « Start », des branches parallèles définies par les nœuds AND-Split et AND-Join et enfin, une boucle définie par le dernier XOR-Split et le premier XOR-Join. □

## 3.3 Aspects de qualité dans les modèles de processus

Les informations de qualité ou les aspects de Qualité de Services (QdS) des modèles de processus sont intuitivement ajoutés par les fournisseurs des modèles. Pour pouvoir les prendre en considération dans le processus de comparaison de modèles de processus, ils sont insérés sous forme d'annotations de qualité dans les graphes de modèles de processus [90].

On peut distinguer deux sortes d'annotations : (i) **une annotation globale** spécifiée sur le graphe du modèle de processus et (ii) **une annotation d'activités**, appelée aussi **annotation locale** [4], définie sur une activité atomique. La définition formelle d'une annotation dans un modèle de processus est la suivante :

4. Une annotation d'activités, appelée aussi annotation locale, définit la valeur d'un aspect de qualité d'une activité donnée, comme, par exemple, son temps d'exécution, sa disponibilité, etc.

**Définition 3.3.1 (Annotation)** Une annotation est représentée par une paire $(m, r)$, où $m$ est un attribut de qualité de services obtenu à partir d'une ontologie $O$ et $r$ est la valeur de l'attribut $m$.

Il est important de noter que certaines propriétés de qualité globales (annotations globales) peuvent être calculées à partir des annotations des activités atomiques, en utilisant des fonctions d'agrégation définies par les fournisseurs de services. Par exemple, le temps d'exécution global d'un modèle de processus peut être calculé en additionnant, pour chaque chemin allant du nœud « Start » au nœud « End », les temps d'exécution des activités atomiques, ensuite la plus grande valeur est choisie. Pour plus de détails sur de telles fonctions, voir [49].

Dans ce travail, nous considérons que les graphes des modèles de processus déjà annotés avec des attributs de qualité. Des techniques permettant d'obtenir ces informations sont discutées dans [49].

**Exemple 3.3.1.** La figure 3.2 présente un exemple de modèles de processus annoté avec des attributs de QdS. Il contient trois annotations globales indiquant le temps de réponse, la fiabilité et la disponibilité du modèle de processus et des annotations d'activités indiquant par exemple le temps de réponse, la fiabilité, la sécurité et le coût de certaines de ses activités. Remarquons que le temps d'exécution global de ce modèle de processus peut être calculé à partir des temps d'exécution des deux branches d'activités atomiques suivantes : (i) « *Login* », « *FindHotel* », « *SelectHotel* » et « *Payment* » et (ii) « *Login* », « *RentApartment* » et « *Payment* ». En effet, dû au nœud connecteur « XOR-Split », le temps d'exécution de ce modèle de processus est égal au maximum entre le temps d'exécution de la branche (i) et le temps d'exécution de la branche (ii). En conséquence, le temps d'exécution de ce modèle de processus est égal au temps d'exécution de la branche la plus lente qui est la branche (ii), i.e., 92ms. □

## 3.4 Intégration de préférences dans une requête utilisateur

Dans une requête, l'utilisateur exprime à la fois la structure du modèle de processus recherché et des contraintes sur les aspects de QdS. Les requêtes utilisateur représentent aussi des graphes de modèles de processus annotés avec des préférences exprimées sur les aspects de qualité [90]. Ces préférences peuvent être soit des **préférences globales** spécifiées au niveau du graphe du modèle de processus requête, soit **des préférences d'activités,** appelée aussi **préférences locales,** formulées au niveau des activités atomiques.

Une préférence peut être définie comme suit :

**Définition 3.4.1 (Préférence)** Une préférence est une expression exprimant le souhait de l'utilisateur sur un aspect de QdS (la fiabilité, la disponibilité, le coût, ...) d'un modèle de processus ou d'une activité.

Deux types de préférences sont considérés : (i) ***les préférences obligatoires***, appelées *contraintes* (*hard preferences*) et (ii) ***les préférences souples***, appelées aussi *souhaits* (*soft preferences*). Les

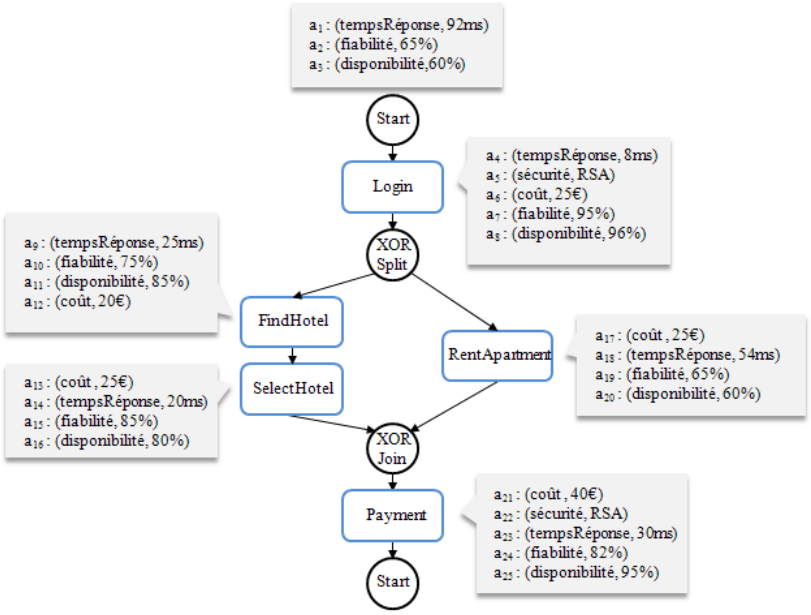



FIGURE 3.2 – Graphe annoté de modèle de processus $t_1$.

contraintes représentent des conditions dont la satisfaction est obligatoire, permettant ainsi la sélection d'un ensemble de modèles de processus candidats. Autrement dit, un modèle de processus ne satisfaisant pas au moins une de ces conditions est éliminé, réduisant ainsi l'espace de recherche. Quant aux préférences souples, elles sont des conditions optionnelles dont la satisfaction n'est pas obligatoire mais plus elles sont satisfaites plus le modèle de processus correspondant est préféré.

Les notions de préférences obligatoires et de préférences souples sont définies par :

**Définition 3.4.2 (Préférence obligatoire)** Une préférence obligatoire est une expression relationnelle de la forme *(m, o, r)*, où $o \in \{<, >, \leq, \geq, =, \neq\}$ est un opérateur relationnel et $r$ est une valeur d'un aspect de qualité de services $m$.

Par exemple, l'expression « (disponibilité, $\geq$, 80%) » est une préférence obligatoire qui signifie que les modèles de processus recherchés doivent avoir une disponibilité d'au moins 80%.

**Définition 3.4.3 (Préférence souple)** Une préférence souple est une expression exprimée à l'aide de constructeurs empruntés au modèle de Kießling [80]. Elle peut prendre l'une des formes suivantes :

- **préférences atomiques :** Dans cette catégorie, six types de préférences souples peuvent être formulés :
  - $around(m, r_{desired}, \mu_{around})$ : pour un attribut $m$, cette expression favorise la valeur $r_{desired}$, sinon, elle favorise celles qui sont proches de $r_{desired}$. La fonction d'appartenance $\mu_{around}$ évalue le degré avec lequel une valeur $r$ satisfait $r_{desired}$ ;
  - $between(m, r_{low}, r_{up}, \mu_{between})$ : pour un attribut $m$, cette expression favorise les valeurs dans l'intervalle $[r_{low}, r_{up}]$, sinon, elle favorise les valeurs proches des bornes. La fonction $\mu_{between}$

évalue le degré avec lequel une valeur $r$ satisfait l'intervalle $[r_{low}, r_{up}]$ ;

- $max(m, \mu_{max})$ : pour un attribut $m$, cette déclaration favorise la valeur la plus élevée, sinon, la valeur la plus proche du maximum est favorisée. Par exemple, le maximum de l'attribut « disponibilité » est égal à 100 %. La fonction $\mu_{max}$ évalue le degré avec lequel une valeur $r$ se rapproche de la valeur maximale de $m$ ;
- $min(m, \mu_{min})$ : pour un attribut $m$, cette expression favorise la valeur la plus petite, sinon, la valeur la plus proche du minimum est favorisée, comme par exemple : le minimum de l'attribut « coût » est égal à 0. $\mu_{min}$ détermine dans quelle mesure une valeur $r$ est proche de la valeur minimale de $m$ ;
- $likes(m, r_{desired})$ : pour un attribut $m$, cette expression favorise la valeur $r_{desired}$, sinon, toute autre valeur est acceptée ;
- $dislikes(m, r_{undesired})$ : pour un attribut $m$, cette expression favorise les valeurs différentes de $r_{undesired}$, sinon, $r_{undesired}$ est acceptée ;

- **préférences complexes :** Deux types de préférences complexes sont étudiées :
  - préférence de Pareto $\otimes(p_i, p_j)$ : cette expression signifie que les deux préférences $p_i$ et $p_j$ sont de même importance ;
  - préférence de priorité $\&(p_i, p_j)$ : cette expression spécifie que la préférence $p_i$ est plus importante que la préférence $p_j$.

D'une manière similaire à [80], les préférences souples atomiques sont classées en : (i) **préférences numériques** exprimées à l'aide des constructeurs « *around* », « *between* », « *max* » et « *min* » et, (ii) **préférences non-numériques** définies en utilisant les constructeurs « *likes* » et « *dislikes* ».

Dans ce travail, nous avons proposé que les préférences définies par l'utilisateur soient caractérisées par des fonctions d'appartenance liées aux préférences atomiques numériques. Ces fonctions permettent de garantir la propriété de commensurabilité [5]. Elles calculent le degré de satisfaction de chaque préférence numérique en tenant compte de la sémantique exprimée par l'utilisateur, prenant ainsi en considération les aspects subjectifs. En effet, pour une même préférence, par exemple $p$ : max(disponibilité) spécifiant un maximum de disponibilité au niveau d'une activité, une annotation d'activité $a$ : (disponibilité, 60) peut être à la fois satisfaisante pour un utilisateur $u_1$, et non satisfaisante pour un autre utilisateur $u_2$, car ce dernier est complètement satisfait lorsque la valeur de l'attribut « disponibilité » est supérieure à 80% et (ii) non satisfait pour une valeur inférieure ou égale à 50%. Quant aux préférences non-numériques, leurs valeurs sont prises à partir d'une ontologie proposée aussi par l'utilisateur.

**Exemple 3.4.1.** La figure 3.3 présente un exemple de requête utilisateur annotée avec trois préférences globales indiquant que l'utilisateur recherche des modèles de processus ayant un temps de réponse inférieur ou égal à 85ms et annoté avec des préférences d'activités portant sur la sécurité, le temps de réponse et le coût des activités atomiques. Les fonctions d'appartenance des préférences

5. La propriété de commensurabilité est vérifiée si et seulement si les degrés de satisfaction résultant de différentes préférences sont comparables et basés sur la même échelle.

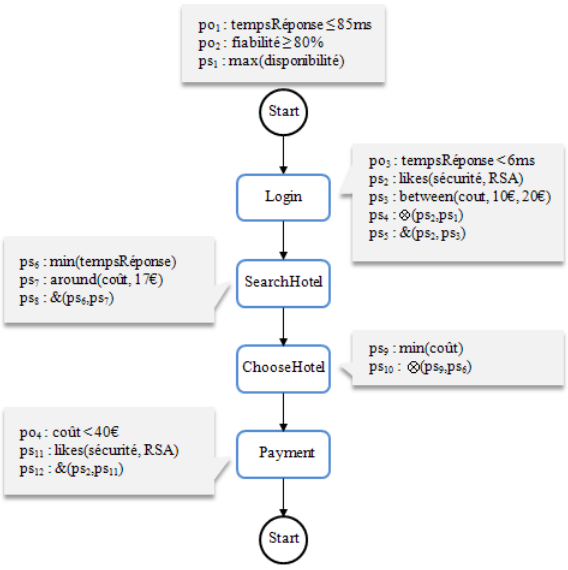



FIGURE 3.3 – Graphe annoté de modèle de processus requête $q_1$.

numériques définies par l'utilisateur sont présentées dans la figure 3.5 page 87. □

## 3.5 Modèle d'évaluation de modèles de processus

Nous discutons, dans cette section, les différentes mesures implémentées dans le système validant l'approche proposée ainsi que les algorithmes correspondants pour évaluer les modèles de processus [8, 9, 6]. Dans un premier temps, nous présentons les méthodes d'évaluation des préférences obligatoires et souples exprimées par l'utilisateur sur les aspects de QdS (i.e., les aspects non-fonctionnels). Nous détaillons également les procédures de calcul pour mesurer la satisfaction globale des préférences de qualité et la similarité structurelle entre deux modèles de processus. Enfin, nous terminons cette sous-section par présenter les méthodes de classement des modèles de processus que nous avons proposées afin d'ordonner les réponses des plus satisfaisantes aux moins satisfaisantes suivant les exigences fonctionnelles et non-fonctionnelles de l'utilisateur. La spécificité de ces mesures réside dans le fait que les degrés retournés ont une sémantique facilement interprétable par l'utilisateur.

Dans ce qui suit, la sous-section 3.5.1 décrit la méthode appliquée dans l'approche de sélection de modèles de processus proposée pour évaluer les préférences obligatoires de l'utilisateur exprimées sur les aspects de qualité. La sous-section 3.5.2 présente la méthode d'évaluation des préférences souples. En sous-section 3.5.3, des mesures pour évaluer la satisfaction globale des préférences non-fonctionnelles, sont proposées. La sous-section 3.5.4 introduit deux mesures permettant de calculer la similarité structurelle entre deux modèles de processus de deux manières différentes. Enfin, la sous-section 3.5.5 présente trois mesures permettant l'évaluation de la similarité globale entre les modèles de processus cibles et un modèle requête en prenant en considération à la fois la satisfaction des préférences utilisateur sur les aspects de QdS et la similarité structurelle.

### 3.5.1 Satisfaction des préférences obligatoires

Dans la méthode de recherche de modèles de processus proposée, les préférences obligatoires sont considérées comme étant des contraintes dont la satisfaction est requise et décisive. Elles jouent le rôle d'un filtre permettant de sélectionner seulement les modèles de processus qui satisfont ces contraintes. Si au moins une d'entre elles est entièrement non satisfaite par un modèle de processus alors celui-ci est écarté et éliminé de l'espace de recherche.

#### 3.5.1.1 Évaluation directe

Les préférences obligatoires sont de deux types comme indiqué dans la section 3.4 : (i) *les préférences obligatoires globales* et (ii) *les préférences obligatoires d'activités* (*locales*, i.e., des préférences liées à des activités). En conséquence, les modèles de processus à sélectionner doivent impérativement *vérifier à la fois toutes les préférences obligatoires globales et toutes les préférences obligatoires locales* exprimées par l'utilisateur.

Considérons, dans ce qui suit, $q$ une requête utilisateur à préférences, $t$ un modèle de processus cible et $M$ l'appariement structurel résultant entre $q$ et $t$. Les fonctions vérifiant les conditions citées ci-dessus sont intégrées comme suit dans le système développé pour évaluer l'approche proposée :

– Pour tester si toutes les préférences obligatoires globales exprimées dans $q$ sont (plus ou moins) satisfaites par les annotations globales de $t$, la mesure représentée par la formule (3.1) est proposée (où le symbole $\wedge$ désigne une conjonction). Dans le cas où l'ensemble des préférences obligatoires globales de $q$ est vide, i.e. $S_{poG}(q) = \emptyset$ , le degré de satisfaction globale de toutes ces préférences, noté $\delta_{poG}(q,t)$, est égal à 1, car le modèle de processus cible $t$ ne viole aucune préférence obligatoire globale de $q$ et toute valeur de ses attributs de QdS est acceptable et satisfaisante. Dans le cas contraire, i.e. $S_{poG}(q) \neq \emptyset$, le degré de satisfaction globale est calculé à partir des degrés de satisfaction élémentaires (ou partiels) obtenus à l'aide des fonctions d'appartenance définies par l'utilisateur explicitant la sémantique des préférences obligatoires globales. Comme toutes ces préférences doivent être plus ou moins satisfaites par l'ensemble $S_{annG}(t)$ des annotations globales de $t$, nous adoptons une vision conjonctive et nous utilisons l'opérateur « min » pour combiner l'ensemble des degrés de satisfaction. Enfin, le degré de satisfaction globale de la requête $q$ comparée à $t$ selon les préférences obligatoires globales est égal à 0 dans tout autre cas (comme, par exemple, lorsqu'il n'existe aucune annotation globale correspondant à au moins une préférence obligatoire globale de la requête).

$$\delta_{poG}(q,t) \quad = \quad \begin{cases} 1, & S_{poG}(q) = \emptyset \\ \bigwedge\limits_{p \in S_{poG}, a \in S_{annG}} (\delta(p,a)) & S_{poG}(q) \neq \emptyset \\ 0, & \text{sinon} \end{cases} \tag{3.1}$$

où $a \in S_{annG}$ est l'annotation globale correspondant à la préférence obligatoire globale $p$

suivant l'appariement $M$ et $\delta(p, a)$ représente le degré de satisfaction de $p$ par rapport à $a$ qui est calculé en appliquant soit la fonction d'appartenance correspondante dans le cas où $p$ est une préférence numérique (voir le tableau 3.2 page 86) soit l'une des deux formules présentées dans le tableau 3.3 page 88 dans le cas où $p$ est une préférence non-numérique.

– Pour tester si toutes les préférences obligatoires locales à une activité requête sont (plus ou moins) satisfaites, nous proposons la mesure exprimée par la formule (3.2). Cette mesure nous permet donc de vérifier si aucune préférence obligatoire d'une activité requête n'a un degré de satisfaction égal à 0. Pour cela, si aucune préférence obligatoire n'est exprimée au niveau d'une activité requête $v$ (i.e., $S_{poA}(v) = \emptyset$), son degré de satisfaction globale $\delta_{poA}(v, w)$ selon la notion de préférences obligatoires d'activités est égal à 1, car toute valeur sur les aspects de QdS d'une activité cible, est acceptable. Dans le cas où l'activité $v$ est mise en correspondance avec l'activité cible $w$ (i.e., $w \neq \$$), l'opérateur « min » (comme dans la formule (3.1)) est utilisé pour agréger l'ensemble des degrés élémentaires. Enfin, dans les autres cas, le degré de satisfaction globale des préférences obligatoires de l'activité $v$, i.e., $S_{poA}(v) \neq \emptyset$, est égal à 0 (comme, par exemple, lorsqu'aucune activité cible n'est mise en correspondance avec l'activité requête $v$ (i.e., $w = \$$), ou encore qu'aucune annotation dans l'activité cible ne correspond à l'une des préférences obligatoires de $v$).

$$\delta_{poA}(v, w) = \begin{cases} 1, & S_{poA}(v) = \emptyset \\ \bigwedge\limits_{p \in S_{poA}, a \in S_{annA}} (\delta(p, a)) & S_{poA}(v) \neq \emptyset \wedge w \neq \$ \\ 0, & \text{sinon} \end{cases} \tag{3.2}$$

où $a \in S_{annA}$ est l'annotation d'activité correspondant à la préférence obligatoire d'activité $p$ suivant l'appariement $M$ et $\delta(p, a)$ représente le degré de satisfaction de $p$ par rapport à $a$.

– Enfin, pour vérifier si toutes les préférences obligatoires (globales et d'activités) de la requête $q$ sont plus au moins satisfaites par les annotations de $t$, la mesure suivante est proposée :

$$\Delta(q, t, M) = \min \left( \delta_{poG}(q, t), \bigwedge_{(v,w) \in M} \delta_{poA}(v, w) \right) \tag{3.3}$$

où le symbole « $\wedge$ » est aussi modélisé par l'opérateur « min ».

**Exemple 3.5.1.** Soient l'appariement structurel $M$ entre $q_1$ (voir la figure 3.3) et $t_1$ (voir la figure 3.2) illustré dans la figure (3.4) et les fonctions d'appartenance trapézoïdales respectives des préférences obligatoires $po_1$, $po_2$, $po_3$ et $po_4$ de $q_1$ : $\mu_{tRep1} = (0, 0, 85, 90)$, $\mu_{fiabilité} = (70, 80, 100, 100)$, $\mu_{tRep2} = (0, 0, 6, 10)$, $\mu_{coût} = (0, 0, 39, 49)$. À partir de l'appariement $M$, nous considérons les paires $(po_i, a_j)$, de sorte que $po_i$ est une préférence obligatoire de la requête $q_1$ et $a_j$ est son annotation correspondante dans $t_1$ : $(po_1, a_1)$, $(po_2, a_2)$, $(po_3, a_4)$, $(po_4, a_{21})$. L'évaluation des préférences obligatoires globales $po_1$ et $po_2$ de $q_1$ suivant la formule (3.1) est donnée par :

$$\delta_{poG}(q_1, t_1) = \min\left(\delta(po_1, a_1), \delta(po_2, a_2)\right) = 0$$

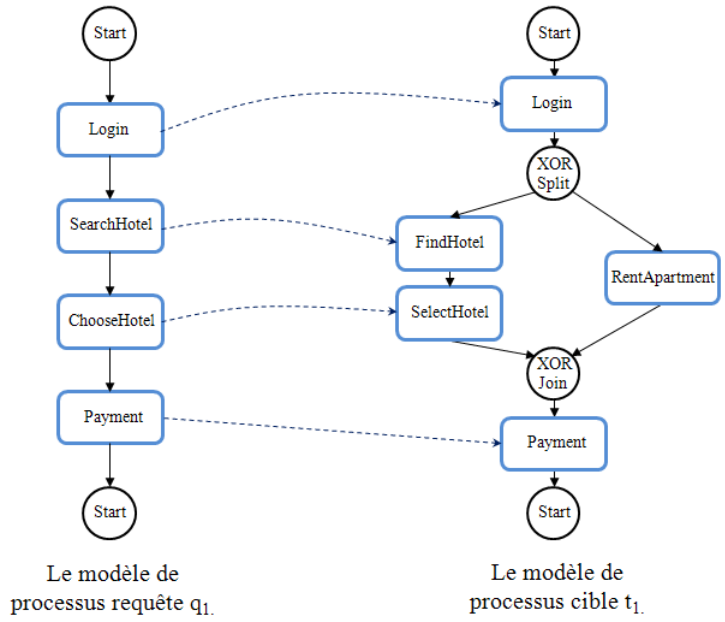



FIGURE 3.4 – Appariement structurel $M$ entre $q_1$ et $t_1$.

Quant à l'évaluation des préférences obligatoires d'activités $po_3$ et $po_4$ de la requête $q_1$, elle donne respectivement :

$$\delta_{poA}(Login, Login) = \delta(po_3, a_4) = 0,5 \text{ et}$$

$$\delta_{poA}(Payment, Payment) = \delta(po_4, a_{21}) = 0,9$$

tel que $\delta_{po_1}(po_1, a_1) = \mu_{tRep1}(92) = 0$, $\delta_{po_2}(po_2, a_2) = \mu_{fiabilité}(65) = 0$, $\delta_{po_3}(po_3, a_4) = \mu_{tRep2}(8) = 0,5$ et $\delta_{po_4}(po_4, a_{21}) = \mu_{coût}(40) = 0,9$ sont les degrés de satisfaction des préférences obligatoires de $q_1$.

Enfin, l'évaluation des préférences obligatoires de $q_1$ suivant la formule (3.3) est donnée par :

$$\Delta(q, t, M) = \min\left(\delta_{poG}(q_1, t_1), \delta_{poA}(Login, Login), \delta_{poA}(Login, Login)\right) = 0$$

Ceci signifie que le modèle de processus $t_1$ ne satisfait pas les préférences obligatoires de $q_1$. Toutefois, en révisant les annotations globales selon les activités de $t_1$ mises en correspondance avec les activités de la requête $q_1$, nous obtenons des résultats concluant que $t_1$ vérifie les préférences obligatoires de $q_1$ (voir l'exemple de la sous-section suivante pour plus d'explications). □

#### 3.5.1.2 Évaluation revisitée

Afin d'améliorer l'évaluation des préférences obligatoires dans le système de sélection de modèles de processus, il est important de déterminer dans quelle étape cette évaluation aura lieu.

Les préférences obligatoires locales d'une requête utilisateur ne peuvent être évaluées tant que

TABLE 3.1 – Degrés de satisfaction des préférences obligatoires de $q_1$ par rapport à $t_1$.

| Préférence obligatoire globale $po_i$ | Annotation globale recalculée $a_i$ | Degré de satisfaction $\delta_{po_i}$ |
|---|---|---|
| $po_1$ | $a_1$ : *(tempsRéponse,83ms)* | 1 |
| $po_2$ | $a_2$ *: (fiabilité, 75%)* | 0,5 |
| $po_3$ | $a_4$ *: (tempsRéponse,8ms)* | 0,5 |
| $po_4$ | $a_{21}$ *: (coût, 40€)* | 0,9 |

les activités cibles correspondantes ne sont pas encore identifiées, car la satisfaction de ces préférences dépend des annotations des activités cibles correspondant. En conséquence, l'évaluation des préférences obligatoires d'activités peut être effectuée après le processus d'appariement structurel mettant en correspondance les activités de la requête avec celles du modèle de processus cible. Toutefois, l'évaluation durant le processus d'appariement permet de réduire l'espace de recherche. En effet, les modèles de processus ne satisfaisant pas un tant soit peu toutes les préférences obligatoires locales sont écartés d'emblée. Pour cela, l'évaluation de ces dernières doit être appliquée lorsque deux activités sont considérées sémantiquement similaires [6].

Les préférences obligatoires globales peuvent être aussi utilisées comme un filtre pour réduire l'espace de recherche, en éliminant les modèles de processus ayant des annotations globales ne satisfaisant pas les préférences obligatoires. Cependant, les annotations globales d'un modèle de processus peuvent être révisées selon l'appariement structurel appliqué entre la requête de l'utilisateur et le modèle de processus cible, en les recalculant à partir des activités cibles mises en correspondance avec les activités de la requête [90].

**Exemple 3.5.2.** À titre illustratif, les valeurs des annotations globales du modèle de processus $t_1$ selon l'appariement $M$ de la figure 3.4, doivent être recalculées à partir des valeurs des annotations de ses activités appariées avec les activités de la requête $q_1$. Ainsi, en considérant les paires d'activités appariées de $M$, le temps de réponse de $t_1$ est égal à 83ms qui est la somme des temps d'exécution des activités appariées de $t_1$, c'est-à-dire, « *Login* », « *FindHotel* », « *SelectHotel* » et « *Payment* ». Quant à la fiabilité et à la disponibilité de $t_1$, elles sont égales respectivement à 75% et 80% qui sont les valeurs minimales respectives des activités appariées de $t_1$.

Par conséquent, en réévaluant les préférences obligatoires de $q_1$ suivant les nouvelles valeurs des annotations globales révisées de $t_1$, nous constatons que $t_1$ satisfait également les préférences obligatoires globales $po_1$ et $po_2$ de $q_1$ avec $\delta_{po_1}(po_1, a_1) = \mu_{tRep1}(83) = 1$ et $\delta_{po_2}(po_2, a_2) = \mu_{fiabilité}(75) = 0,5$. Le tableau 3.1 résume les degrés de satisfaction des préférences obligatoires de la requête $q_1$ après avoir recalculé les annotations globales de $t_1$ suivant l'appariement $M$. □

Pour évaluer les préférences globales d'une requête, il est donc nécessaire de réviser les annotations globales pour déterminer les nouvelles valeurs de qualité du modèle de processus cible en tenant compte de la mise en correspondance de ses activités avec celles de la requête. Pour ce faire,

6. Suivant Gater *et al.* [59], deux activités mises en correspondance sont sémantiquement similaires si la distance entre leur nom, leurs entrées et leurs sorties, n'excède pas un certain seuil défini par l'utilisateur.

la fonction d'évaluation des préférences obligatoires globales devrait être exécutée après chaque appariement structurel entre une requête utilisateur et un modèle de processus cible.

En résumé, pour évaluer la satisfaction des préférences obligatoires d'une requête $q$ appariée avec un modèle de processus $t$, la procédure à trois étapes suivante est appliquée :

1. **Évaluation des préférences obligatoires d'activités**

   Durant le processus d'appariement structurel entre $q$ et $t$, une activité requête $v$ n'est mise en correspondance avec une activité cible $w$ que lorsque toutes les préférences obligatoires d'activités de $v$ sont plus au moins satisfaites par les annotations de qualité de $w$. De cette façon, la puissance du filtrage de l'algorithme d'appariement [59] utilisé dans l'approche proposée est améliorée, en vérifiant si toutes les préférences obligatoires d'une activité requête $v$ sont plus ou moins satisfaites par les annotations de qualité d'une activité cible $w$. Cette vérification peut être réalisée à l'aide de la formule (3.2).

   En d'autres termes, la paire $(v, w)$ n'est ajoutée à l'ensemble $M$ des paires d'activités mises en correspondance durant le processus d'appariement structurel, que lorsque $v$ et $w$ sont sémantiquement similaires et toutes les préférences obligatoires de $v$ sont plus ou moins satisfaites par les annotations de qualité de $w$. Ainsi, le processus d'appariement structurel étendu avec la fonction d'évaluation des préférences obligatoires d'activités, retourne comme résultat un appariement $M$ entre $q$ et $t$ de telle sorte que toutes les préférences obligatoires d'activités de $q$ sont satisfaites par les annotations d'activités de $t$.

2. **Révision des annotations globales de qualité**

   Après l'étape d'appariement structurel entre $q$ et $t$, les annotations globales doivent être recalculées à partir des informations de qualité des activités de $t$ mises en correspondance avec les activités de la requête $q$ comme illustré par l'exemple 3.5.2.

3. **Évaluation des préférences obligatoires globales**

   Enfin, la formule (3.1) est appliquée pour vérifier si toutes les préférences obligatoires globales de $q$ sont plus ou moins satisfaites par les annotations globales de $t$. Ainsi, seuls les modèles de processus structurellement similaires à la requête $q$ et satisfaisant toutes les préférences obligatoires sont sélectionnés.

Nous abordons, dans la sous-section suivante, la méthode d'évaluation des préférences souples (globales et locales) exprimées dans la requête de l'utilisateur sur les aspects non-fonctionnels des modèles de processus recherchés.

### 3.5.2 Satisfaction des préférences souples

Dans le système de sélection de modèles de processus, les préférences souples que nous avons introduites sont des conditions optionnelles dont la satisfaction n'est pas obligatoire, mais plutôt désirable. Plus ces préférences sont satisfaites meilleurs sont les modèles de processus sélectionnés.

Elles interviennent dans le classement des modèles de processus ayant la même fonctionnalité que la requête (i.e., structurellement similaires avec le graphe requête) et satisfaisant les préférences obligatoires de l'utilisateur.

Nous introduisons dans ce qui suit une méthode d'évaluation des préférences souples fondée sur la théorie des ensembles flous. En particulier, nous proposons une mesure, dite **degré de satisfaction ($\delta(q,t)$)**, qui exprime dans quelle mesure les annotations de qualité du modèle de processus cible $t$ satisfont les préférences souples de la requête $q$ selon l'appariement $M$.

#### 3.5.2.1 Évaluation directe

L'évaluation de la satisfaction des préférences souples nécessite une méthode permettant de mesurer la satisfaction des préférences souples atomiques tout en prenant en considération leurs importances exprimées à travers des préférences souples complexes. Pour cela, nous présentons dans ce qui suit un modèle d'évaluation des préférences atomiques et des préférences souples complexes.

– **Préférences souples atomiques**

Comme les préférences souples atomiques peuvent être numériques ou non numériques, nous discutons dans ce qui suit deux approches d'évaluation :

**a) Préférences atomiques numériques**

Le degré de satisfaction des préférences atomiques numériques est calculé en utilisant les fonctions d'appartenance définies par l'utilisateur. Chaque fonction permet d'évaluer dans quelle mesure une annotation donnée, $a : (m, r)$, satisfait la contrainte floue décrite par une préférence $p$ sur le même attribut de qualité $m$.

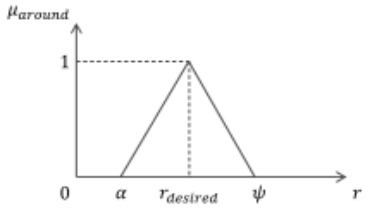
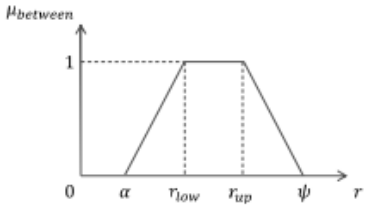
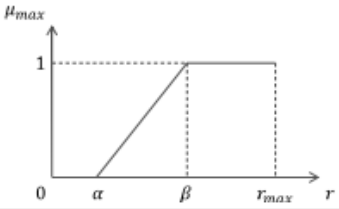
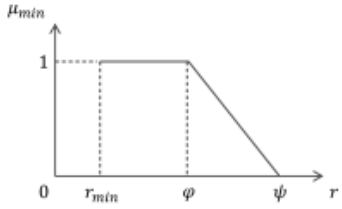
À titre d'exemple, une préférence $p$ de la forme $p : between(m, r_{low}, r_{up})$ est caractérisée par la fonction d'appartenance représentée par un trapèze $(\alpha, \beta, \varphi, \psi)$, où $\beta = r_{low}$, $\varphi = r_{up}$ et $\alpha$ et $\psi$ sont deux valeurs du domaine de l'attribut concerné par la préférence $p$, i.e. $m$. Pour une annotation $a : (m, r)$ d'un modèle de processus cible, le degré de satisfaction de la préférence $p$ par rapport à $a$, noté $\delta(p, a)$, est donné par :

– $p$ est complètement satisfaite si et seulement si $r \in [r_{low}, r_{up}] : \delta(p, a) = 1$ ;
– plus la valeur de $r$ est inférieure à $r_{low}$ ou supérieure à $r_{up}$, moins $p$ est satisfaite : $0 < \delta(p, a) < 1$ ;
– pour $r \in ]-\infty, \alpha] \cup [\psi, +\infty[$, la préférence $p$ n'est pas du tout satisfaite : $\delta(p, a) = 0$.

Nous résumons dans le tableau 3.2 page suivante, la modélisation floue des préférences numériques basées sur les constructeurs présentés dans la définition (3.4.3) de la section 3.4 page 76.

**Exemple 3.5.3.** Considérons l'appariement $M$ de la figure 3.4 et les fonctions d'appartenance des préférences souples atomiques numériques $ps_1$, $ps_3$, $ps_6$, $ps_7$ et $ps_9$ de la requête $q_1$, illustrées dans la figure 3.5. Après avoir révisé les annotations globales, la disponibilité du modèle de processus $t_1$ est évaluée à 80%. Par conséquent, le degré de satisfaction de la préférence souple globale (atomique numérique) $ps_1$ par rapport à l'annotation $a_3$ est : $\delta_{ps_1} = \delta(ps_1, a_3) = \mu_{max}(80) = 1$. Concernant les

TABLE 3.2 – Modélisation floue des préférences atomiques numériques.

| *Préférence numérique p* | *Fonction d'appartenance* | *Exemple* |
|---|---|---|
| $around(m, r_{desired}, \mu_{around})$ | $\mu_{around}(r) = \begin{cases} 0, & r \leq \alpha, r \geq \psi \\ \frac{r-\alpha}{r_{desired}-\alpha}, & \alpha < r < r_{desired} \\ 1, & r = r_{desired} \\ \frac{\psi-r}{\psi-r_{desired}}, & r_{desired} < r < \psi \end{cases}$   | Considérons une préférence $p: around(tempsRéponse, 20ms)$ et sa fonction d'appartenance $\mu_{around}[10ms, 20ms, 20ms, 30ms]$. Cette fonction exprime que l'annotation $a: (tempsRéponse, 18ms)$ est satisfaisante à un degré de 0.8. |
| $between(m, r_{low}, r_{up}, \mu_{between})$ | $\mu_{between}(r) = \begin{cases} 0, & r \leq \alpha, r \geq \psi \\ \frac{r-\alpha}{r_{low}-\alpha}, & \alpha < r < r_{low} \\ 1, & r_{low} \leq r \leq r_{up} \\ \frac{\psi-r}{\psi-r_{up}}, & r_{up} < r < \psi \end{cases}$   | Considérons une préférence $p: between(coût, 20€, 30€)$ et sa fonction d'appartenance $\mu_{between}[18€, 20€, 30€, 33€]$. Cette fonction décrit que l'annotation $a: (coût, 19€)$ est satisfaisante à un degré égal à 0.5. |
| $max(m, \mu_{max})$ | $\mu_{max}(r) = \begin{cases} 0, & r \leq \alpha \\ \frac{r-\alpha}{\beta-\alpha}, & \alpha < r < \beta \\ 1, & \beta \leq r \leq r_{max} \end{cases}$   | Considérons une préférence $p: max(fiabilité)$ et sa fonction d'appartenance $\mu_{max}[70\%, 90\%, 90\%, 100\%]$. Cette fonction décrit que l'annotation $a: (fiabilité, 65\%)$ ne satisfait pas $p$. |
| $min(m, \mu_{min})$ | $\mu_{min}(r) = \begin{cases} 1, & r_{min} \leq r \leq \varphi \\ \frac{\psi-r}{\psi-\varphi}, & \alpha < r < \psi \\ 0, & r \geq \psi \end{cases}$   | Considérons une préférence $p: min(coût)$ et sa fonction d'appartenance $\mu_{min}[20€, 20€, 30€, 35€]$. Cette fonction décrit que l'annotation $a: (coût, 34€)$ est satisfaisante à un degré égal à 0.2. |

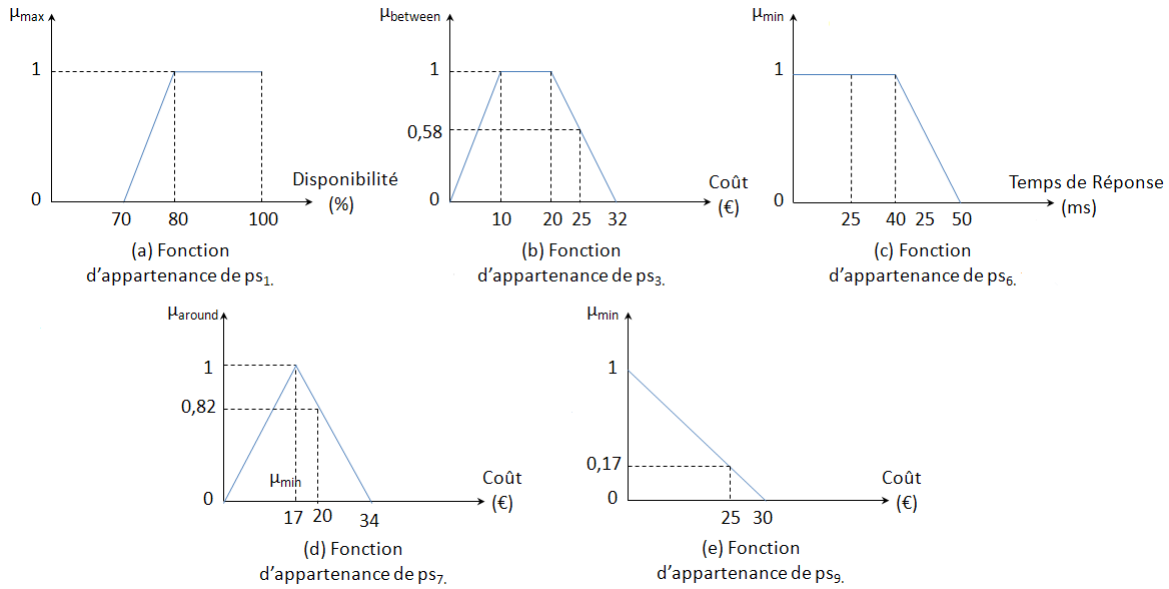



FIGURE 3.5 – Fonctions d'appartenance des préférences numériques de $q_1$.

préférences souples d'activités (atomiques numériques) $ps_3$, $ps_6$, $ps_7$ et $ps_9$, leurs degrés de satisfaction sont respectivement : $\delta_{ps_3} = \delta(ps_3, a_6) = \mu_{between}(25) = 0,58$, $\delta_{ps_6} = \delta(ps_6, a_9) = \mu_{min}(25) = 1$, $\delta_{ps_7} = \delta(ps_7, a_{12}) = \mu_{around}(20) = 0,82$ et $\delta_{ps_9} = \delta(ps_9, a_{13}) = \mu_{min}(25) = 0,17$. □

**b) Préférences atomiques non-numériques**

Le degré de satisfaction des préférences atomiques de type non-numérique est évalué en utilisant une méthode de calcul de similarité sémantique entre concepts. Par exemple, pour une préférence construite à l'aide du constructeur « *likes* », sa satisfaction est obtenue en évaluant la similarité sémantique entre le concept souhaité et le concept défini dans l'annotation correspondante. Parmi les méthodes proposées dans la littérature, nous utilisons la similarité sémantique développée dans [141], qui est définie comme suit :

Considérons une ontologie, notée $O$, et deux concepts notés respectivement $c_1$ et $c_2$. La similarité sémantique, $wp$, entre $c_1$ et $c_2$ est donnée par :

$$wp(O, c_1, c_2) = {}^{2N_3}/_{(N_1+N_2+2N_3)} \tag{3.4}$$

où $c_3$ est le super-concept le plus proche de $c_1$ et de $c_2$, $N_1$ est la longueur du chemin (en nombre d'arêtes) entre $c_1$ et $c_3$, $N_2$ est la longueur du chemin entre $c_2$ et $c_3$ et, $N_3$ est la longueur du chemin entre $c_3$ et la racine de l'ontologie $O$.

Étant donné une préférence atomique non numérique $p$ et une annotation $a$, le degré de satisfaction $\delta(p, a)$ entre $a$ et $p$ est calculé comme présenté dans le tableau 3.3.

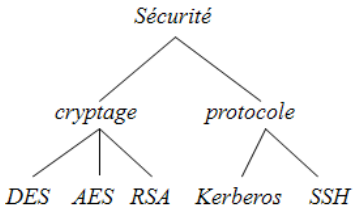



FIGURE 3.6 – Ontologie des préférences atomiques non-numériques sur l'attribut de QdS « Sécurité ».

TABLE 3.3 – Degré de satisfaction d'une préférence atomique non-numérique $p$ suivant une annotation $a : (m, r)$.

| Préférence non-numérique $p$ | Degré de satisfaction $\delta(p, a)$ |
|---|---|
| $likes(m, r_{desired})$ | $\delta(p,a) = \begin{cases} 1, & \text{si } r_{desired} = r \\ wp(O, r_{desired}, r), & \text{sinon} \end{cases}$ |
| $dislikes(m, r_{undesired})$ | $\delta(p, a) = 1 - \delta(likes(m, r_{undesired}), a)$ |

**Exemple 3.5.4.** À partir de l'appariement structurel $M$ entre $q_1$ et $t_1$ et l'ontologie de sécurité de la figure 3.6 extraite de [72], on déduit que les degrés $\delta_2$ et $\delta_{11}$ de satisfaction des préférences atomiques non-numériques $ps_2$ et $ps_{11}$ avec leurs annotations respectives $a_5$ et $a_{22}$ sont : $wp(O, RSA, RSA) = \delta(ps_2, a_5) = \delta(ps_{11}, a_{22}) = 1$, i.e., $\delta_{ps_2} = \delta_{ps_{11}} = 1$. □

– **Préférences souples complexes**

Les préférences complexes ne sont en réalité qu'une façon d'exprimer la priorité ou l'importance d'une préférence par rapport à une autre. Pour évaluer l'ensemble de ces préférences, nous avons donc besoin de les interpréter et de les transformer en préférences pondérées tout en respectant le point de vue de l'utilisateur. Pour cela, nous construisons l'arbre, dit *arbre de préférences* $t_p$, qui représente la sémantique des préférences complexes perçue par l'utilisateur.

Dans cet arbre, les nœuds représentent les préférences atomiques composant les préférences complexes, et les arêtes définissent une relation de type « *plus important que* » (i.e. une préférence de priorité &) du nœud parent au nœud fils. Les nœud de même niveau ayant le même nœud parent expriment des préférences de Pareto, notées ⊗. À chaque niveau $i$ de l'arbre $t_p$, est associé un poids $\omega_i = 1/i$, excepté pour $i = 0$ (i.e., la racine). Ce poids représente donc l'importance de l'utilisateur vis-à-vis de chaque préférence du niveau $i$ de l'arbre $t_p$. Dans l'algorithme 3.1 page suivante, les étapes de construction de l'arbre de préférences $t_p$ sont décrites [92].

**Exemple 3.5.5.** Considérons l'arbre de préférences, $t_p$, de la requête $q_1$ illustré dans la figure 3.7. L'arbre $t_p$ est formé des préférences atomiques $ps_1$, $ps_2$, $ps_3$, $ps_6$, $ps_7$, $ps_9$ et $ps_{11}$, et permet d'exprimer les préférences complexes $ps_4$, $ps_5$, $ps_8$, $ps_{10}$ et $ps_{12}$. Par exemple, la préférence complexe $ps_4$, $\otimes(ps_2, ps_1)$ de type Pareto, est composée des préférences atomiques $ps_2$ et $ps_1$ et signifie que $ps_2$ et $ps_1$ sont de même importance. La préférence complexe $ps_5$, $\&(ps_2, ps_3)$ préférence à priorité, est composée de $ps_2$ et de $ps_3$ et signifie que la préférence atomique $ps_2$ est plus importante que $ps_3$.

**Algorithme 3.1** Construction de l'arbre de préférences $t_p$.

**Entrée :** L'ensemble des préférences souples $S_{ps} = S_{ps}^A \cup S_{ps}^C$
$S_{ps}^A = \{ps_1, ..., ps_l\}$ : les préférences atomiques
$S_{ps}^C = \{ps_{l+1}, ..., ps_n\}$ : les préférences complexes
**Sortie :** L'arbre de préférences $t_p$.
**Début**
Soit $t_p$ un arbre
// Création des nœuds de $t_p$ à partir des préférences souples atomiques
**Pour chaque** préférence atomique $ps_i \in S_{ps}^A$ **faire** créer le nœud $n_i$ dans $t_p$
//Création des arêtes de $t_p$ à partir des préférences à priorité (&)
**Pour chaque** préférence complexe $ps_k \in S_{ps}^C$ tel que $ps_k : \&(ps_i, ps_j)$ **faire**
créer l'arête $(n_i, n_j)$ dans $t_p$
//Préservation de la cohérence des préférences complexes dans l'arbre $t_p$
**Pour chaque** arête $(n_i, n_j) \in t_p$ tel qu'il existe un autre chemin de $n_i$ à $n_j$ **faire**
supprimer l'arête $(n_i, n_j)$ // Exemple de tel cas : $\&(ps_i, ps_k)$, $\&(ps_i, ps_j)$ et $\&(ps_j, ps_k)$

//Attribution du niveau d'importance 1 aux nœuds $n_i$ des préférences à priorité $\&(ps_i, ps_j)$
**Pour chaque** nœud $n_i \in t_p$ n'ayant pas de nœud parent **faire** $n_i.niveau \leftarrow 1$

Soit $N$ l'ensemble des nœuds de $t_p$ ayant un niveau d'importance égal à 1
//Attribution des niveaux d'importance suivants aux nœuds fils $n_j$
**Pour chaque** nœud $n_i \in N$ **faire**
**Pour chaque** arête $(n_i, n_j)$ **faire**
$n_j.niveau \leftarrow n_i.niveau + 1$
$N \leftarrow N \cup \{n_j\}$

// Réévaluation des niveaux d'importance à partir des préférences de Pareto
**Pour chaque** préférence complexe $ps_k \in S_{ps}^C$ tel que $ps_k : \otimes(ps_i, ps_j)$ **faire**
//Cas où $ps_j$ ne figure pas dans une préférence à priorité
**Si** $n_i.niveau \neq null$ **et** $n_j.niveau = null$ **alors** $n_j.niveau \leftarrow n_i.niveau$
//Cas où $ps_i$ ne figure pas dans une préférence à priorité
**Sinon Si** $n_i.niveau = null$ **et** $n_j.niveau \neq null$ **alors** $n_i.niveau \leftarrow n_j.niveau$
//Mettre $n_i$ et $n_j$ au niveau 1 de l'arbre $t_p$ sinon
**Sinon Si** $n_i.niveau = null$ **et** $n_j.niveau = null$ **alors** $n_i.niveau \leftarrow 1$ ; $n_j.niveau \leftarrow 1$
**Sinon Si** $n_i.niveau \neq null$ et $n_j.niveau \neq null$ **alors**
**Si** $n_i.niveau < n_j.niveau$ **alors**
$n_j.niveau \leftarrow n_i.niveau$
mettre à jour les descendants de $n_j$
**Sinon Si** $n_j.niveau < n_i.niveau$ **alors**
$n_i.niveau \leftarrow n_j.niveau$
mettre à jour les descendants de $n_i$

//Insertion des préférences atomiques ne figurant pas dans des préférences complexes
**Pour chaque** nœud $n_i$ n'ayant pas de niveau d'importance **faire** $n_i.niveau \leftarrow 1$

Retourner $t_p$

**Fin**

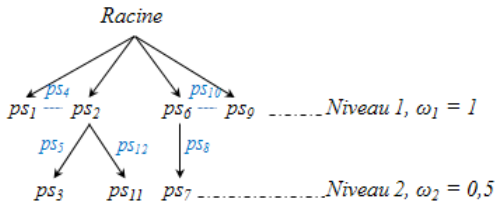



FIGURE 3.7 – Arbre de préférences $t_p$ de la requête $q_1$.

En appliquant ainsi l'algorithme 3.1, nous obtenons les poids suivants :

- $\omega_1 = 1$ correspondant au poids d'importance des préférences atomiques $ps_1$, $ps_2$, $ps_6$, et $ps_9$
- $\omega_2 = 0,5$ correspondant au poids d'importance des préférences atomiques $ps_3$, $ps_7$ et $ps_{11}$.

Ces poids traduisent le niveau d'importance de chaque préférence atomique vis-à-vis de l'utilisateur. □

Pour prendre en considération les poids d'importance dans l'évaluation des préférences atomiques, nous calculons, comme défini dans [45], un nouveau degré de satisfaction, noté $\delta'_i$, pour chaque préférence atomique $p_i$ présente dans les préférences complexes exprimées par l'utilisateur. Pour cela, la formule suivante est appliquée :

$$\delta'_i = max\left(\delta_i, 1 - \omega_i\right) \tag{3.5}$$

où $\omega_i$, avec $\max\limits_{i=1,\ldots,n} \omega_i = 1$, dénote le poids d'importance d'une préférence atomique $p_i$ et $\delta_i$ est son degré de satisfaction calculé soit à l'aide d'une fonction d'appartenance dans le cas d'une préférence numérique (voir le tableau 3.2), soit en utilisant une des deux formules, présentées dans le tableau 3.3, dans le cas contraire.

Cette nouvelle interprétation de la préférence $p_i$ considère que toute valeur en dehors du support de sa fonction d'appartenance est acceptable à un degré égal à $1-\omega_i$. Cela signifie que plus grand est $\omega_i$ (i.e., $p_i$ est importante), plus petit est le degré de satisfaction d'une valeur en dehors du support de $p_i$. En outre, la formule (3.5) exprime aussi que plus $\omega_i$ est grand (i.e., plus $p_i$ est importante), plus la satisfaction de $p_i$ est prise en compte (i.e., $\delta_i$).

**Exemple 3.5.6.** Reprenons les degrés de satisfaction des préférences souples atomiques obtenus dans les exemples 3.5.3 et 3.5.4 et leurs poids d'importance respectifs résultant de l'arbre de préférence $t_p$ de la figure 3.7. Par application de la formule (3.5) sur les préférences souples atomiques de $q_1$, la nouvelle interprétation $\delta'_i$ des degrés de satisfaction de ces préférences atomiques suivant les annotations correspondantes dans $t_1$, est décrite dans le tableau 3.4. □

#### 3.5.2.2 Évaluation revisitée

Comme pour les préférences obligatoires, les préférences souples sont de deux types : (i) les préférences souples globales et (ii) les préférences souples locales (i.e., associées à des activités). L'évaluation des préférences souples locales dépend de l'appariement structurel entre la requête et

TABLE 3.4 – Degrés de satisfaction des préférences atomiques de $q_1$ par rapport à $t_1$.

| Préférence atomique $ps_i$ | Poids d'importance $\omega_i$ | Degré de satisfaction $\delta_{ps_i}$ | Degré de satisfaction $\delta'_{ps_i}$ |
|---|---|---|---|
| $ps_1$ | 1 | 1 | 1 |
| $ps_2$ | 1 | 1 | 1 |
| $ps_3$ | 0,5 | 0,58 | 0,58 |
| $ps_6$ | 1 | 1 | 1 |
| $ps_7$ | 0,5 | 0,82 | 0,82 |
| $ps_9$ | 1 | 0,17 | 0,17 |
| $ps_{11}$ | 0,5 | 1 | 1 |

les modèles de processus cibles et, l'évaluation des préférences souples globales dépend aussi de la mise en correspondance des activités cibles avec celles de la requête.

Pour évaluer la satisfaction globale des préférences souples d'une requête $q$ par rapport aux annotations de qualité d'un modèle de processus cible $t$, nous procédons comme suit :

1. **Évaluation des préférences souples atomiques**

   À partir d'un appariement structurel $M$ entre $q$ et $t$, nous évaluons le degré de satisfaction de chaque préférence souple atomique. Pour cela, nous utilisons leurs fonctions d'appartenance définies par l'utilisateur dans le cas des préférences atomiques numériques et appliquons l'équation 3.4 page 87 dans le cas où les préférences souples atomiques évaluées sont non-numériques.

2. **Évaluation des préférences souples complexes**

   En deuxième étape, les préférences souples complexes sont évaluées pour déterminer le poids d'importance de chaque préférence souple atomique. Pour ce faire, nous construisons d'abord l'arbre de préférences correspondant en utilisant l'algorithme 3.1 présenté précédemment. Ensuite, nous appliquons la formule (3.5) pour recalculer les degrés de satisfaction des préférences souples atomiques en prenant en compte leur poids d'importance vis-à-vis de l'utilisateur.

3. **Satisfaction globale des préférences souples**

   Enfin, une fonction d'agrégation est appliquée pour évaluer le degré de satisfaction globale des préférences souples exprimées par l'utilisateur. Dans cette thèse, nous proposons trois fonctions d'agrégation définies dans la sous-section suivante.

### 3.5.3 Satisfaction globale des préférences

Afin d'évaluer la satisfaction globale des préférences utilisateur exprimées sur les aspects de qualité d'un modèle de processus cible, nous proposons dans ce travail les trois mesures suivantes : (i) *la mesure basée sur la moyenne* ; (ii) *la mesure basée sur les quantificateurs linguistiques* et (iii) *la mesure fondée sur les conditions bipolaires*. Chacune de ces mesures associe une sémantique différente

au degré de satisfaction globale des préférences. Elles reçoivent en entrée l'ensemble des degrés de satisfaction des préférences obligatoires $S_{po} = \{\delta_{po_1}, ... , \delta_{po_m}\}$ et l'ensemble des degrés de satisfaction des préférences souples $S_{sp} = \{\delta_{ps_1}, ..., \delta_{ps_n}\}$ et retournent comme résultat un degré de satisfaction globale $\delta_{pref}$ des préférences d'une requête utilisateur $q$ par rapport aux annotations d'un modèle de processus cible $t$.

#### 3.5.3.1 Mesure basée sur la moyenne arithmétique

Pour évaluer le degré de satisfaction globale $\delta_{pref}$ des préférences de qualité d'une requête, cette mesure calcule la moyenne entre les degrés de satisfaction des préférences obligatoires $\{\delta_{po_1}, ..., \delta_{po_m}\}$ et les degrés de satisfaction des préférences souples $\{\delta_{ps_1}, ..., \delta_{ps_n}\}$ comme suit :

$$\delta_{pref}(q,t) = \frac{\sum_{i=1}^{m} \delta_{po_i} + \sum_{i=1}^{n} \delta_{ps_i}}{m+n} \tag{3.6}$$

La formule (3.6) considère que le degré de satisfaction des préférences souples $\delta_{sp}$ précède le degré de satisfaction des préférences obligatoires $\delta_{hp}$, c'est-à-dire, lorsqu'une préférence obligatoire est définie sur le même attribut de qualité qu'une préférence souple, au niveau d'une activité ou d'un modèle de processus, le degré de satisfaction de la préférence souple est celui pris en considération dans ce calcul de moyenne. Les préférences obligatoires représentent dans ce cas un filtre et les préférences souples déterminent des contraintes plus raffinées et plus spécifiques. Toutefois, le degré de satisfaction globale $\delta_{pref}$ résultant de la formule (3.6) est difficile à interpréter par l'utilisateur.

**Exemple 3.5.7.** Reprenons les degrés de satisfaction des préférences globales et d'activités de $q_1$ obtenus dans les exemples 3.5.1, 3.5.2 et 3.5.6. Par application de la formule (3.6), la satisfaction globale des préférences de QdS de la requête $q_1$ comparées aux annotations du modèle de processus cible $t_1$ est calculée comme suit :

$\delta_{pref}(q_1, t_1) = \frac{(\delta_{po_1}+\delta_{po_2}+\delta_{po_3}+\delta_{po_4})+(\delta_{ps_1}+\delta_{ps_2}+\delta_{ps_3}+\delta_{ps_6}+\delta_{ps_7}+\delta_{ps_9}+\delta_{ps_{11}})}{4+7}$

Ainsi, le degré de satisfaction des préférences de QdS de $q_1$ est : $\delta_{pref}(q_1, t_1) = 0,77$. □

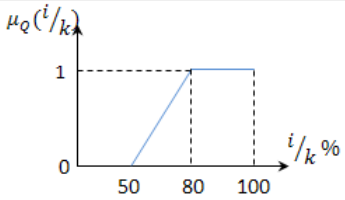
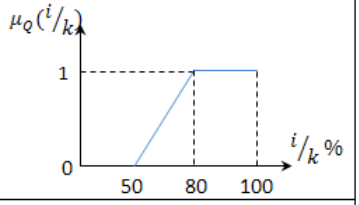
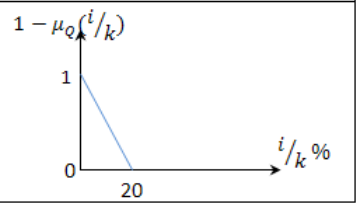
#### 3.5.3.2 Mesure basée sur les quantificateurs linguistiques

Une interprétation de nature flexible de la satisfaction des préférences d'une requête $q$ par un modèle de processus cible $t$ peut être définie par le degré de vérité de la proposition suivante :

$\gamma_1$ : presque toutes les préférences souples de $q$ sont satisfaites par $t$.

La proposition ci-dessus est une expression quantifiée floue de la forme « $Q$ $X$ sont $P$ », où (i) $Q$ est un quantificateur relatif [61] qui est défini par une fonction $\mu_Q$ telle que $\mu_Q(\varpi)$ est le degré de vérité de « $Q$ $X$ sont $P$ » quand une proportion $\varpi$ d'éléments de $X$ satisfait plus ou moins le prédicat $P$ et les autres éléments n'étant pas satisfaits ; (ii) $X$ est un ensemble d'éléments ; (iii) $P$ est un prédicat flou. Dans [142], une méthode par décomposition calculant le degré de vérité $\delta_\gamma$ de « $\gamma : Q$ $X$ sont $P$ » est proposée (voir la section 2.2 du chapitre 2).

TABLE 3.5 – Interprétations basées sur la décomposition des propositions $\gamma_1$, $\gamma_2$ et $\gamma_3$.

| *Proposition* $\gamma_i$ | *Ensembles X et Ω* | *Fonction d'appartenance* $\mu_Q(^i/_k)$ |
|---|---|---|
| $\gamma_1$ | $X = \{sp_1, \ldots, sp_k\}$ : l'ensemble des préférences souples atomiques de la requête $q$.<br>$\Omega = \{\mu_1 : \delta_{ps_1}, \ldots, \mu_k : \delta_{ps_k}\}$ : l'ensemble des degrés de satisfaction des préférences souples. |  |
| $\gamma_2$ | $X = \{(v_1, w_1), \ldots, (v_k, w_k)\}$ : l'ensemble des paires d'activités appariées entre la requête $q$ et la cible $t$.<br>$\Omega = \{\mu_1 : ss(v_1, w_1), \ldots, \mu_k : ss(v_k, w_k)\}$ : l'ensemble des degrés de similarité sémantique des activités appariées. |  |
| $\gamma_3$ | $X = \{(v_1, w_1), \ldots, (v_k, w_k)\}$ : l'ensemble des paires d'activités appariées entre la requête $q$ et la cible $t$.<br>$\Omega = \{\mu_1 : c(v_1, w_1), \ldots, \mu_k : c(v_k, w_k)\}$ : l'ensemble des coûts de transformation des activités $v_i$ en activités $w_i, 1 \leq i \leq k$. |  |

Considérons $X$ l'ensemble des préférences souples (i.e., les préférences souples globales et locales) d'une requête utilisateur $q$ et $\mu_Q$ la fonction d'appartenance du quantificateur relatif « *presque tous* ». La sémantique de ce dernier est illustrée dans le tableau 3.5. Elle exprime que : (i) l'utilisateur est totalement satisfait si au moins 80% des préférences sont satisfaites et (ii) l'utilisateur n'est pas du tout satisfait si au plus 50% des préférences sont satisfaites.

En se basant sur la méthode par décomposition de Yager [142] décrite dans la section 2.2 du chapitre 2, le degré de vérité $\delta_{\gamma_1}$ de la proposition $\gamma_1$ est calculé comme suit :

$$\delta_{\gamma_1} = \max_{1 \leq i \leq n} \min \left( \mu_i, \mu_Q \left( {}^i/_n \right) \right) \tag{3.7}$$

où $\Omega = \{\mu_1 : \delta_{ps_1}, \ldots \mu_n : \delta_{ps_n}\}$ est l'ensemble des degrés de satisfaction des préférences souples de $q$ classés dans un ordre décroissant, i.e. $\mu_1 \geq \ldots \geq \mu_n$, et $\delta_{ps_i}$ dénote le degré de satisfaction d'une préférence souple atomique $p_i$ tel que présenté dans la sous-section 3.5.2.

La formule (3.7) calcule le degré de satisfaction globale des préférences d'une requête utilisateur $q$ par rapport aux annotations d'un modèle de processus cible $t$, i.e., $\delta_{pref}(q, t) = \delta_{\gamma_1}$. Dans le cas où $\mu_Q({}^i/_n) = {}^i/_n$ dans la formule 3.7, ce degré véhicule une sémantique simple et claire vis-à-vis de l'utilisateur [96]. Ainsi, l'évaluation de $\gamma_1$ signifie que « *au moins* $\delta^*_{pref}$% *des préférences de* $q$ *sont satisfaites par* $t$ *à au moins un degré de* $\delta_{pref}$ », où $\delta^*_{pref} = 100 \times \delta_{pref}$.

**Exemple 3.5.8.** Considérons les degrés de satisfaction des préférences souples, $\delta'_{ps_i}$, de la requête $q_1$ présentés dans le tableau (3.4). La fonction d'appartenance du quantificateur linguistique « presque tous » de la proposition $\gamma_1$ est $\mu_Q$, définie par $\mu_Q({}^i/_n) = {}^i/_n$. En se basant sur l'approche par décomposition de Yager [142], le degré de vérité de la proposition « $\gamma_1$ : presque toutes les préfé-

rences souples de $q_1$ sont satisfaites par $t_1$ », permet d'obtenir le degré de satisfaction des préférences de $q_1$ par rapport aux annotations de $t_1$, en procédant comme suit :

- les degrés de satisfaction des préférences souples sont d'abord ordonnés dans l'ordre décroissant : $\{\mu_1 : \delta_{ps_1} = 1,\ \mu_2 : \delta_{ps_2} = 1,\ \mu_3 : \delta_{ps_6} = 1,\ \mu_4 : \delta_{ps_{11}} = 1,\ \mu_5 : \delta_{ps_7} = 0,82,\ \mu_6 : \delta_{ps_3} = 0,58,\ \mu_7 : \delta_{ps_9} = 0,17\}$.
- Ensuite, la formule (3.7) est appliquée :

$$\delta_{pref} = \max\left(\min\left(1, \mu_Q\left(1/7\right)\right), \min\left(1, \mu_Q\left(2/7\right)\right), ..., \min\left(0,17, \mu_Q\left(7/7\right)\right)\right) = 0,71$$

Ce degré signifie qu'*au moins 71% des préférences de la requête $q_1$ sont satisfaites par $t_1$ à au moins un degré de 0,71*. En effet, il y a cinq préférences parmi sept qui ont un degré de satisfaction supérieur ou égal à 0,82, ce qui est donc supérieur ou égal à 0,71. □

#### 3.5.3.3 Mesure basée sur la bipolarité

La mesure basée sur la notion de bipolarité permet d'évaluer les préférences bipolaires des requêtes utilisateur de la forme « trouver les modèles de processus dans lesquels *toutes les préférences obligatoires sont satisfaites et si possible au moins une préférence souple est satisfaite* ». Pour ce faire, chaque préférence obligatoire $po_i$ est associée à une préférence atomique souple $ps_i$ exprimées sur le même attribut de qualité $m_i$ au niveau de l'activité ou du modèle de processus. Chaque couple de préférences formé $(po_i, ps_i)$ représente ainsi une préférence bipolaire de la forme : « *préférence obligatoire $po_i$ et si possible préférence souple $ps_i$* », où $i$ indique que les deux préférences sont exprimées sur le même attribut de QdS. Toutefois, des cas particuliers peuvent se produire pour certaines conditions bipolaires :

- *Absence de préférence souple.* Dans ce cas, la préférence bipolaire est interprétée par « préférence obligatoire $po_i$ et si possible préférence obligatoire $po_i$ » [98]. Cela signifie que la préférence souple est la même que la préférence obligatoire, i.e., $ps_i = po_i$.
- *Absence de préférence obligatoire.* Dans ce cas, $po_i = true$, c'est-à-dire, toutes les réponses sont acceptables et si possible celles satisfaisant la préférence souple $ps_i$ [48].

**Exemple 3.5.9.** En se basant sur l'appariement $M$ entre $q_1$ (Figure 3.3) et $t_1$ (Figure 3.2), les préférences bipolaires formées sont : $(po_1, po_1)$, $(po_2, po_2)$, $(po_3, po_3)$, $(po_4, po_4)$, $(true, ps_1)$, $(true, ps_2)$, $(true, ps_3)$, $(true, ps_6)$, $(true, ps_7)$, $(true, ps_9)$ et $(true, ps_{11})$. Dans les quatre premières préférences bipolaires, chacune des préférences obligatoires, $po_j$, ne possède pas de préférence souple correspondante (i.e., sur le même aspect de qualité). Ainsi, le premier cas cité ci-dessus est appliqué pour chaque paire bipolaire contenant une préférence obligatoire. Quant aux sept dernières paires de préférences, aucune préférence obligatoire ne correspond à une préférence souple, $ps_i$, suivant les mises en correspondance établies par $M$. Pour cela, le deuxième cas cité ci-dessus est appliqué pour chaque préférence souple de la requête de telle sorte que la préférence obligatoire de chaque paire est remplacée par la valeur « *true* ». □

Dans le cadre de cette approche, le degré de satisfaction globale $\delta_{pref}$ des préférences bipolaires d'une requête $q$ est représenté par un couple de degrés bipolaires, $\delta_{pref} = (\delta_{S_{po}}, \delta_{S_{ps}})$, qui signifie que les annotations cibles satisfont (i) *toutes les préférences obligatoires à au moins un degré de* $\delta_{S_{po}}$, et (ii) *au moins une préférence souple à au moins un degré de* $\delta_{S_{ps}}$ ». Pour calculer ces degrés, deux étapes sont nécessaires :

1. En premier lieu, pour chaque préférence bipolaire $(po_i, ps_i)$, la paire des degrés bipolaires $(\delta_{po_i}, \delta_{ps_i})$, $i = 1, ..., k$, est formée à partir des ensembles des degrés de satisfaction $S_{S_{po}}$ et $S_{S_{ps}}$. Pour les cas particuliers discutés ci-dessus, $\delta_{ps_i} = \delta_{po_i}$ dans le cas où il n'existe pas de préférence souple $ps_i$, et $\delta_{po_i} = 1$ lorsqu'il n'y a pas de préférence obligatoire $po_i$.
2. En deuxième lieu, la paire des degrés bipolaires $(\delta_{S_{po}}, \delta_{S_{ps}})$ entre une requête et un modèle de processus est déterminée en appliquant la formule (2.15) de la section 2.2 sur les paires des degrés bipolaires $\{(\delta_{po_1}, \delta_{ps_1}), ..., (\delta_{po_k}, \delta_{ps_k})\}$, c'est-à-dire :

$$\delta_{pref} = (\delta_{S_{po}}, \delta_{S_{ps}}) = \left( \min_{1 \leq i \leq k} \delta_{po_i}, \min \left( \min_{1 \leq i \leq k} \delta_{po_i}, \max_{1 \leq i \leq k} \delta_{ps_i} \right) \right)$$

   Ainsi, la paire des degrés bipolaires $(\delta_{S_{po}}, \delta_{S_{ps}})$ exprime dans quelle mesure l'ensemble des degrés de satisfaction des préférences utilisateur satisfait *toutes les préférences obligatoires et si possible au moins une préférence souple.*

**Exemple 3.5.10.** Considérons les paires de préférences obtenues dans l'exemple 3.5.9. Pour évaluer la satisfaction des préférences de la requête $q_1$, les degrés de satisfaction des paires bipolaires sont d'abord construits à partir des degrés de satisfaction des préférences obligatoires et souples présentés dans les tableaux 3.1 et 3.4 : $(1, 1)$, $(0,5, 0,5)$, $(0,5, 0,5)$, $(0,9, 0,9)$, $(1, 1)$, $(1, 1)$, $(1, 0,58)$, $(1, 1)$, $(1, 0,82)$, $(1, 0,17)$ et $(1, 1)$. Enfin, en appliquant la formule (2.2) de la section 2.2, le degré de satisfaction des préférences bipolaires est : $\delta_{pref} = (0,5, 0,5)$. Ce degré bipolaire signifie que « *toutes les préférences obligatoires sont satisfaites à au moins un degré de 0,5 et au moins une préférence souple est satisfaite à au moins un degré de 0,5* ». □

### 3.5.4 Similarité structurelle

Pour évaluer la similarité structurelle entre le graphe d'un modèle de processus requête $q$ et le graphe d'un modèle de processus cible $t$, nous utilisons les résultats de l'algorithme d'appariement structurel entre $q$ et $t$, à savoir, l'ensemble $E$ des opérations d'édition nécessaires pour transformer $q$ en $t$ et l'appariement $M$ contenant l'ensemble des paires $(v, w)$ tel que $v$ est une activité de $q$ et $w$ est une activité de $t$. Tel que présenté dans la sous-section 3.5.1 page 80, pour chaque paire d'activités appariées $(v, w) \in M$, $v$ et $w$ sont sémantiquement similaires et satisfont toutes les préférences obligatoires d'activités.

Ainsi, le degré de similarité structurelle $\delta_{SS}$ entre $q$ et $t$ est calculé au moyen de l'une des trois mesures présentées ci-dessous, où chacune d'entre elles associe une sémantique différente au degré $\delta_{SS}$. Elles reçoivent donc en entrée l'appariement $M$ et l'ensemble $E$, et retourne le degré de similarité structurelle $\delta_{SS}$.

En sous-section 3.5.4.1, une mesure basée sur les quantificateurs linguistiques pour évaluer le degré $\delta_{SS}$ est proposée. Cette mesure permet d'évaluer la similarité structurelle entre deux graphes de modèles de processus en prenant en considération les contraintes structurelles définies par l'utilisateur, qui considère qu'un modèle de processus cible est intéressant par rapport à sa requête si : « *les activités cibles sont appariées avec presque toutes les activités de la requête et la requête ne nécessite presque aucune opération d'édition pour la transformer en ce modèle de processus cible* ». Dans la sous-section 3.5.4.2, nous présentons deux mesures basées sur la notion de bipolarité pour évaluer la similarité structurelle entre deux graphes de modèles de processus. La première mesure bipolaire calcule le degré de similarité structurelle $\delta_{SS}$ suivant les exigences structurelles de l'utilisateur, en considérant qu'un modèle de processus cible est intéressant par rapport à la requête si : « *les activités cibles sont appariées avec presque toutes les activités de la requête* et si possible *la requête ne nécessite presque aucune opération d'édition pour la transformer en ce modèle de processus cible* ». Enfin, la deuxième mesure bipolaire considère qu'un modèle de processus cible est une réponse intéressante par rapport aux contraintes structurelles de l'utilisateur si : « *la similarité entre les activités mises en correspondance est maximale* et si possible *le coût de transformation est minimal* ».

#### 3.5.4.1 Mesure basée sur les quantificateurs linguistiques

La méthode basée sur les quantificateurs linguistiques présentée dans la sous-section 3.5.3.2 est également appliquée pour évaluer le degré de similarité structurelle entre deux modèles de processus. La similarité structurelle entre un modèle de processus requête $q$ et un modèle de processus cible $t$ peut être donnée par le degré de vérité des propositions suivantes :

$\gamma_2$ : presque toutes les activités de $q$ sont appariées avec les activités de $t$, et
$\gamma_3$ : presque aucune opération d'édition n'est nécessaire pour transformer $q$ en $t$.

Pour évaluer le degré de vérité de la proposition $\gamma_2$, nous considérons $X$ l'ensemble des paires d'activités mises en correspondance entre $q$ et $t$ et $\mu_Q$ la fonction d'appartenance du quantificateur relatif « *presque tous* ». La sémantique de ce dernier est illustrée dans le tableau 3.5. Dans ce cas, (i) l'utilisateur est totalement satisfait si au moins 80% des activités requêtes sont appariées et (ii) l'utilisateur n'est pas du tout satisfait si seulement au plus 50% des activités requêtes sont appariées.

En se basant sur la méthode par décomposition de Yager [142], le degré de vérité $\delta_{\gamma_2}$ de la proposition $\gamma_2$ est calculé comme suit :

$$\delta_{\gamma_2} = \max_{1 \leq i \leq n} \min \left( \mu_i, \mu_Q \left( {}^{i}/_{n} \right) \right) \tag{3.8}$$

où $\Omega = \{\mu_i : ss(v_1, w_1), ...\mu_n : ss(v_n, w_n)\}$ est l'ensemble des degrés de similarité sémantique des activités requêtes appariées tels que $\mu_1 \geq ... \geq \mu_n$ et $ss(v_i, w_i)$, pour $i = 1, ..., n$, dénote le degré de similarité sémantique d'une activité requête $v_i$ mise en correspondance avec une activité cible $w_i$.

La formule (3.8) calcule donc le degré de similarité sémantique globale des activités de la requête $q$ appariées avec les activités d'un modèle de processus cible $t$. Dans cette thèse, nous considérons en particulier la formule (3.8) dans le cas où $\mu_Q({}^{i}/_{n}) = {}^{i}/_{n}$, car de cette façon les degrés de vérité

obtenus ont une sémantique simple et claire pour l'utilisateur [96]. Ainsi, l'évaluation de $\gamma_2$ signifie que « *au moins* $\delta^*_{\gamma_2}\%$ *des activités de* $q$ *sont mises en correspondance avec les activités de* $t$ *à au moins un degré de* $\delta_{\gamma_2}$ », où $\delta^*_{\gamma_2} = 100 \times \delta_{\gamma_2}$.

Concernant la proposition $\gamma_3$, l'expression « presque aucune opération d'édition n'est nécessaire pour transformer $q$ en $t$ » est équivalente à l'expression « *es les opérations d'édition ne sont pas nécessaires pour transformer* $q$ *en* $t$ ». En conséquence, le degré de vérité de $\gamma_3$ est calculé comme suit :

$$\delta_{\gamma_3} = \max_{1 \leq i \leq n} \min \left(1 - \mu_i, 1 - \mu_Q \left(i/n\right)\right) \tag{3.9}$$

Dans ce cas, $\Omega = \{\mu_1 : c(v_1, w_1), ..., \mu_n : c(v_n, w_n)\}$ est l'ensemble des coûts normalisés de transformation des activités requêtes appariées avec les activités cibles, $c(v_i, w_i)$, pour $i = 1, ..., n$, représente le coût normalisé de transformation [7] d'une activité requête $v_i$ en une activité cible $w_i$, $X$ est l'ensemble $M$ des paires d'activités mises en correspondance entre $q$ et $t$ et $\mu_Q$ la fonction d'appartenance du quantificateur relatif « *presque tous* ». La sémantique de ce dernier est illustrée dans le tableau 3.5. Dans ce cas, (i) l'utilisateur est totalement satisfait si la requête $q$ ne nécessite aucune opération d'édition et (ii) l'utilisateur n'est pas du tout satisfait si au moins 20% de la structure de la requête $q$ nécessitent des opérations d'édition pour la transformer en la cible $t$.

De la même manière, en considérant le cas où $\mu_Q(i/n) = i/n$ dans la formule (3.8), le degré de vérité obtenu a une sémantique simple et claire [96], qui signifie que « *au moins* $\delta^*_{\gamma_3}\%$ *de la structure de* $q$ *ne nécessite pas d'opérations d'édition pour transformer* $q$ *en* $t$ *à un au moins un degré de* $\delta_{\gamma_3}$ », où $\delta^*_{\gamma_3} = 100 \times \delta_{\gamma_2}$.

Enfin, la similarité structurelle entre $q$ et $t$ est évaluée comme suit :

$$\delta_{SS}(q, t) = \min \left(\delta_{\gamma_2}, \delta_{\gamma_3}\right) \tag{3.10}$$

La formule (3.10) calcule donc le degré de similarité structurelle $\delta_{SS}$ entre un modèle de processus requête $q$ et un modèle de processus $t$ en prenant en considération les exigences structurelles de l'utilisateur exprimées dans les propositions $\gamma_2$ et $\gamma_3$. De cette façon, le degré $\delta_{SS}$ devient facilement interprétable par l'utilisateur, qui signifie que « *au moins* $\delta^*_{SS}\%$ *des activités requêtes sont appariées avec les activités de* $t$ *à au moins un degré de* $\delta_{SS}$ *et au moins* $\delta^*_{SS}\%$ *de* $q$ *ne nécessite pas de transformation à au moins un degré de* $\delta_{SS}$ ».

**Exemple 3.5.11.** Considérons l'appariement structurel $M$ établi entre $q_1$ et $t_1$ présenté dans la figure 3.4 page 82. Supposons maintenant que les degrés de similarité sémantique et les coûts de transformation entre les nœuds mis en correspondance dans $M$ sont ceux présentés dans le tableau 3.6. Ainsi, le degré de similarité structurelle globale entre $q_1$ et $t_1$ est calculé en deux étapes. La première étape consiste à calculer les degrés de vérité des propositions $\gamma_2$ et $\gamma_3$ en appliquant

7. Le coût normalisé de transformation d'un nœud $n_1$ en un nœud $n_2$ est $c(n_1, n_2) = c(E')/1+c(E')$, où $(n_1, n_2) \in M$, $E'$ l'ensemble des opérations d'édition de $E$ appliquée sur $n_1$ et $c(E')$ est la somme des coûts des opérations d'édition de $E'$ [92].

TABLE 3.6 – Degrés de similarité sémantique et coûts de transformation des nœuds appariés de $q_1$.

| Nœuds appariés $(v_i, w_i)$ | Similarité sémantique $ss(v_i, w_i)$ | Coût de transformation $c(v_1, w_1)$ |
|---|---|---|
| (Login, Login) | 1 | 0 |
| (SearchHotel, FindHotel) | 0,75 | 0,54 |
| (ChooseHotel, SelectHotel) | 0,78 | 0,54 |
| (Payment, Payment) | 1 | 0 |

respectivement les formules (3.8) et (3.9) comme suit :

$\delta_{\gamma_2}(q_1, t_1) = \max(\min(1, \mu_Q(1/4)), ..., \min(0,75, \mu_Q(4/4))) = 0,75$ et

$\delta_{\gamma_3}(q_1, t_1) = \max(\min(1 - 0,54, 1 - \mu_Q(1/4)), ..., \min(1 - 0, 1 - \mu_Q(4/4))) = 0,5.$

La deuxième étape est de calculer la similarité structurelle en utilisant la formule (3.10). Ainsi, le degré de similarité entre $q_1$ et $t_1$ suivant l'appariement $M$ est $\delta_{SS}(q_1, t_1) = \min(0,75, 0,5) = 0,5$. Ce résultat signifie qu'*au moins 50% des activités de la requête $q_1$ sont appariées avec $t_1$ à au moins un degré de 0,5 et, au moins 50% de la structure de $q_1$ ne nécessite d'être transformée en $t_1$ à au moins un degré de 0,5.* □

#### 3.5.4.2 Mesures bipolaires

Les mesures basées sur la notion de bipolarité, proposées dans cette sous-section, permettent d'évaluer la similarité structurelle entre un modèle de processus requête et un modèle de processus cible, en prenant en considération les exigences structurelles de l'utilisateur exprimées sous forme de conditions floues bipolaires. Relativement aux objectifs des utilisateurs, la similarité structurelle peut être interprétée de différentes façons lors de la recherche de modèles de processus. Par exemple, la similarité structurelle peut être exprimée par l'une des conditions bipolaires suivantes :

1. « trouver un modèle de processus cible $t$ avec lequel *presque toutes les activités requêtes sont appariées* (contrainte) *et si possible aucune opération d'édition n'est nécessaire pour transformer la requête $q$ en $t$* (souhait) ».
2. « trouver un modèle de processus cible $t$ ayant *une similarité sémantique maximale* (contrainte) *et si possible nécessitant un coût de transformation minimal* (souhait) ».

Dans la première condition bipolaire, les deux expressions floues définies à l'aide des quantificateurs linguistiques, i.e., la contrainte et le souhait, ont été déjà étudiées dans la sous-section 3.5.4.1. Ainsi, les degrés de vérité de la contrainte et du souhait sont respectivement $\delta_{\gamma_2}$ et $\delta_{\gamma_3}$ (voir les formules (3.8) et (3.9)). Le degré de similarité structurelle $\delta_{SS}$ entre $q$ et $t$ suivant la première condition bipolaire est un couple de degrés défini comme suit :

$$\delta_{SS}(q, t) = (\delta_{\gamma_2}, \min(\delta_{\gamma_2}, \delta_{\gamma_3})) \tag{3.11}$$

Cette formule vérifie la propriété de cohérence qui permet d'exprimer qu'un souhait ne peut être préféré plus qu'il ne soit accepté.

Concernant la deuxième condition bipolaire, l'évaluation du degré de similarité structurelle entre une requête $q$ et un modèle de processus cible $t$ nécessite d'abord l'évaluation de leur similarité sémantique ainsi que le coût des opérations d'édition nécessaires pour transformer $q$ en $t$. Pour ce faire, considérons $M$ un appariement structurel entre $q$ et $t$ et $E$ l'ensemble des opérations d'édition nécessaires pour transformer $q$ en $t$.

La similarité sémantique entre $q$ et $t$ est définie par la moyenne des similarités sémantiques $ss(v_i, w_i)$ des paires d'activités mises en correspondance ( $(v_i, w_i) \in M$), c'est-à-dire :

$$ss(q,t) = \frac{\sum_{(v,w)\in M} ss(v,w)}{|M|} \tag{3.12}$$

Quant au coût de transformation entre $q$ et $t$, noté $c(q,t)$, il est obtenu en additionnant les coûts des opérations d'édition de $E$, comme le montre la formule ci-dessous :

$$c(q,t) = \sum_{o\in E} c(o) \tag{3.13}$$

où $c(o)$ représente le coût de l'opération d'édition $o$ de $E$.

Ainsi, la similarité structurelle entre $q$ et $t$ suivant la deuxième condition bipolaire est aussi un couple de degrés calculé comme suit :

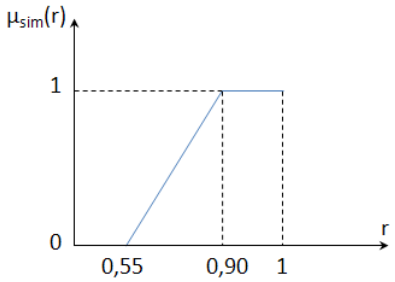
$$\delta_{SS}(q,t) = (\mu_{sim}(ss(q,t)), \min(\mu_{sim}(ss(q,t)), \mu_{coût}(c(q,t)))) \tag{3.14}$$

où $\mu_{sim}$ et $\mu_{coût}$ représentent respectivement les fonctions d'appartenance définies par l'utilisateur pour interpréter la sémantique des termes *maximale* et *minimal* exprimés dans la deuxième condition bipolaire.

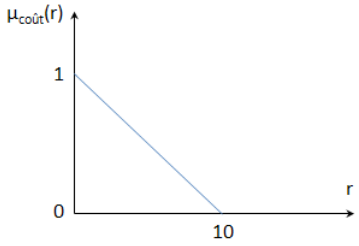
L'évaluation de chacune des deux conditions bipolaires présentées dans cette sous-section permet de calculer la similarité structurelle $\delta_{SS}$ sous forme de deux degrés $\delta_{SS} = (\gamma_c, \gamma_w)$, tels que $\gamma_c$ est le degré de satisfaction de la contrainte et $\gamma_w$ est le degré de satisfaction du souhait.

**Exemple 3.5.12.** Considérons les degrés de vérité $\delta_{\gamma_2}$ et $\delta_{\gamma_3}$ évalués dans l'exemple 3.5.11. Par application de la formule (3.11), la similarité structurelle entre $q_1$ et $t_1$ suivant la première condition bipolaire est donnée par $\delta_{SS}(q_1, t_1) = (0,75, 0,5)$. Ce couple de degrés a pour sémantique « *au moins 75% des activités de $q_1$ sont appariées avec les activités de $t_1$ à au moins un degré de 0,75 et au moins 50% de la structure de $q_1$ ne nécessite d'être transformée en $t_1$ à au moins un degré de 0,5* ».

Considérons maintenant les degrés de similarité sémantique et les coûts de transformation présentés dans le tableau (3.6) et les fonctions d'appartenance de la figure (3.8) interprétant les expressions : « *une similarité sémantique maximale* » et « *un coût de transformation minimal* ». Pour évaluer la similarité structurelle entre $q_1$ et $t_1$ suivant la deuxième condition bipolaire, les degrés de similarité sémantique et de coût de transformation sont d'abord évalués en utilisant respectivement les formules (3.12) et (3.13). Par conséquent, la similarité sémantique et le coût de transformation entre




(a) Fonction d'appartenance de « similarité sémantique maximale ».




(b) Fonction d'appartenance de « coût de transformation minimal ».

FIGURE 3.8 – Fonctions d'appartenance des expressions : « *une similarité sémantique maximale* » et « *un coût de transformation minimal* ».

$q_1$ et $t_1$ sont respectivement $ss(q_1, t_1) = 0,88$ et $c(q_1, t_1) = 1,08$. Enfin, l'évaluation de la deuxième condition bipolaire entre $q_1$ et $t_1$ est donné par :

$\delta_{SS}(q_1, t_1) = (\mu_{sim}(0,88), \min(\mu_{ss}(0,88), \mu_{coût}(1,08)) = (0,94, 0,94)$.

Ainsi, le résultat obtenu a pour sémantique « *au moins 94% des activités de la requête $q_1$ sont appariées et au moins 94% de la structure de $q_1$ ne nécessite pas d'être transformée en $t_1$* ». □

### 3.5.5 Similarité globale et ordonnancement

Afin d'ordonner les modèles de processus candidats des plus satisfaisants aux moins satisfaisants par rapport à une requête, il est donc important d'évaluer leur similarité globale en prenant en considération les contraintes fonctionnelles et non-fonctionnelles de l'utilisateur. Pour cela, quatre mesures d'agrégation sont proposées dans cette sous-section. Elles prennent en entrée le degré de similarité structurelle $\delta_{SS}$ et le degré de satisfaction globale des préférences de qualité $\delta_{pref}$, pour déterminer enfin le degré de similarité globale $\delta_{SG}(q, t)$ entre la requête $q$ et le modèle de processus cible $t$.

#### 3.5.5.1 Mesure basée sur la moyenne pondérée

Cette mesure permet de combiner en un seul degré $\delta_{SG}$, le degré de similarité structurelle $\delta_{SS}$ et le degré de satisfaction globale des préférences de qualité $\delta_{pref}$. Pour cela, l'utilisateur définit le poids d'importance $\omega_{SS} \in [0, 1]$ de la similarité structurelle. Ainsi, la similarité globale entre $q$ et $t$ est calculée par la moyenne pondérée de $\delta_{SS}$ et de $\delta_{pref}$ définie comme suit :

$$\delta_{SG}(q, t) = \omega_{SS} \times \delta_{SS} + (1 - \omega_{SS}) \times \delta_{pref} \tag{3.15}$$

De cette façon, l'ensemble des modèles de processus candidats à la requête de l'utilisateur peuvent être ordonnés du plus satisfaisant au moins satisfaisant par rapport aux contraintes structurelles et de qualité de l'utilisateur.

**Exemple 3.5.13.** Considérons le degré de satisfaction des préférences $\delta_{pref}(q_1, t_1) = 0,71$ et le

degré de similarité structurelle $\delta_{SS}(q_1, t_1) = 0,5$ évalués, respectivement, dans les exemples 3.5.8 et 3.5.11. Supposons que l'utilisateur considère que la similarité est plus importante que la satisfaction des préférences de qualité avec un degré $\omega_{SS} = 0,75$. Par application de la formule (3.15), le degré de similarité globale entre $q_1$ et $t_1$ est donné par :

$\delta_{SG}(q_1, t_1) = 0,75 \times 0,5 + 0,25 \times 0,71 = 0,55$. □

#### 3.5.5.2 Mesure basée sur la min-combinaison

Pour évaluer la similarité globale $\delta_{SG}$ entre le modèle de processus cible $t$ et la requête $q$, la mesure basée sur la min-combinaison [48] utilise l'opérateur « min » entre le degré de similarité structurelle $\delta_{SS}$ et le degré de satisfaction globale des préférences de qualité $\delta_{pref}$, c'est-à-dire,

$$\delta_{SG}(q, t) = \min(\delta_{SS}, \delta_{pref}) \tag{3.16}$$

Ainsi, la plus petite valeur des deux degrés $\delta_{SS}$ et $\delta_{pref}$ est sélectionnée. Cela signifie que les contraintes fonctionnelles et non-fonctionnelles de l'utilisateur sont satisfaites à au moins un degré de $\delta_{SG}$. À l'aide de cette mesure, les modèles de processus candidats sont ordonnés dans l'ordre croissant de leur similarité globale avec la requête de l'utilisateur.

**Exemple 3.5.14.** Reprenons le degré de satisfaction des préférences $\delta_{pref}(q_1, t_1) = 0,71$ et le degré de similarité structurelle $\delta_{SS}(q_1, t_1) = 0,5$ calculés respectivement dans les exemples 3.5.8 et 3.5.11. En appliquant la formule (3.16) de la mesure basée sur la min-combinaison, le degré de similarité globale entre $q_1$ et $t_1$ est donné par :

$\delta_{SG}(q_1, t_1) = \min(0,71, 0,5) = 0,5$.

Ce degré signifie que « *le modèle de processus $t_1$ est structurellement similaire à $q_1$ à au moins un degré de 0,5 et $t_1$ satisfait les préférences de QdS de $q_1$ à au moins un degré de 0,5* ». □

#### 3.5.5.3 Mesures basées sur des conditions bipolaires

La similarité globale entre un modèle de processus $t$ et une requête $q$ peut également avoir une interprétation bipolaire. Par exemple, l'utilisateur peut définir la similarité globale par l'une des deux conditions bipolaires suivantes :

1. « trouver un modèle de processus dans lequel *toutes les contraintes sont satisfaites* (la contrainte) *et si possible au moins un souhait est satisfait* (le souhait) ». L'évaluation de cette condition est réalisée au moyen de la mesure d'agrégation basée sur les opérateurs « min » et « max », i.e.,

$$\delta_{SG}(q, t) = (c_{SG}, w_{SG}) = \left(\min\left(\gamma_c, \delta_{S_{po}}\right), \max\left(\gamma_w, \delta_{S_{ps}}\right)\right) \tag{3.17}$$

où les paires des degrés $(\gamma_c, \gamma_w)$ et $(\delta_{S_{po}}, \delta_{S_{ps}})$ représentent respectivement le degré de similarité structurelle $\delta_{SS}$ et le degré de satisfaction globale des préférences de qualité $\delta_{pref}$, i.e., $\delta_{SS} =$

$(\gamma_c, \gamma_w)$ et $\delta_{pref} = (\delta_{S_{po}}, \delta_{S_{ps}})$ tel que présenté respectivement dans les sous-sections 3.5.4.2 et 3.5.3.3.

Pour ordonner les modèles de processus candidats des plus intéressants aux moins intéressants par rapport à la requête $q$, un ordre lexicographique peut être appliqué. Les modèles de processus sont d'abord ordonnés selon « $c_{SG}$ » et lorsque deux modèles ont la même valeur $c_{SG}$, le degré $w_{SG}$ est utilisé pour les départager.

2. « trouver un modèle de processus ayant *une similarité structurelle maximale et si possible satisfaisant les préférences obligatoires et si possible satisfaisant les préférences souples* ». L'évaluation de la similarité globale $\delta_{SG}$ entre $q$ et $t$ à partir de cette condition bipolaire *hiérarchique,* retourne un vecteur de trois degrés :

$$\delta_{SG}(q,t) = (\delta_{SS}, \delta'_{S_{po}}, \delta'_{S_{ps}}) = \left(\delta_{SS}, \min\left(\delta_{SS}, \delta_{S_{po}}\right), \min\left(\delta_{SS}, \delta_{S_{po}}, \delta_{S_{ps}}\right)\right) \tag{3.18}$$

Cette mesure est appliquée lorsque l'utilisateur considère que (i) la similarité structurelle $\delta_{SS}$ est plus importante que la satisfaction des préférences obligatoires $\delta_{S_{po}}$ et des préférences souples $\delta_{S_{ps}}$, et (ii) la satisfaction des préférences obligatoires $\delta_{S_{po}}$ est plus importante que la satisfaction des préférences souples $\delta_{S_{ps}}$.

À l'aide de cette mesure, les modèles de processus peuvent être ordonnés par l'application de l'ordre lexicographique en considérant la similarité structurelle $\delta_{SS}$ comme étant le critère le plus prioritaire, ensuite la satisfaction des préférences obligatoires $\delta'_{S_{po}}$ comme critère secondaire, enfin la satisfaction des préférences souples $\delta'_{S_{ps}}$ comme critère tertiaire pour départager les ex-æquo.

**Exemple 3.5.15.** Considérons les degrés bipolaires de la satisfaction des préférences de QdS et de la similarité structurelle, $\delta_{pref}(q_1, t_1) = (0,5, 0, 5)$ et $\delta_{SS}(q_1, t_1) = (0, 94, 0, 94)$, obtenus respectivement dans les exemples 3.5.10 et 3.5.12. Le degré de similarité globale entre $q_1$ et $t_1$ suivant la première condition bipolaire présentée ci-dessus, est calculé comme suit :

$\delta_{SG}(q_1, t_1) = (\min(0, 94, 0, 5), \max(0, 94, 0, 5)) = (0, 5, 0, 94)$.

Ce résultat a pour sémantique : « *au moins* 50% *des contraintes de la requête* $q_1$ *sont satisfaites à un degré de* $0, 5$ *et au moins* 94% *des souhaits de* $q_1$ *sont satisfaits à au moins un degré de* $0, 94$ ». □

## 3.6 Méthodologie d'évaluation des modèles de processus

Nous présentons, dans cette section, la méthodologie d'évaluation appliquée par l'approche proposée pour sélectionner des modèles de processus en prenant en considération les exigences fonctionnelles (i.e., la structure) et non-fonctionnelles (i.e. les informations de QdS) des utilisateurs. Dans un premier temps, nous définissons l'architecture de la méthodologie d'évaluation et ses trois principales procédures, à savoir la procédure de calcul de similarité structurelle, la procédure d'évaluation des préférences de qualité et celle d'évaluation de la similarité globale entre un modèle de processus

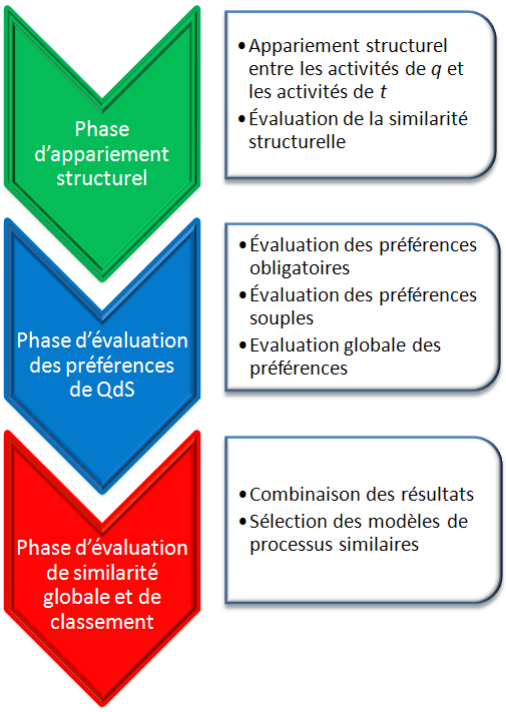



FIGURE 3.9 – Principales phases de la méthodologie de sélection de modèles de processus.

cible et une requête utilisateur. Par la suite, nous décrivons toutes les phases de la méthodologie proposée et les différentes étapes requises pour la sélection de modèles de processus satisfaisant une requête à préférence. Enfin, nous conclurons cette section par une discussion et un bilan résumant les principales contributions de l'approche d'évaluation de modèles de processus proposée.

### 3.6.1 Architecture du système de sélection de modèles de processus

L'objectif étant de sélectionner les modèles de processus respectant les exigences fonctionnelles et non-fonctionnelles des utilisateurs, nous proposons une méthodologie composée de trois procédures principales (voir la figure 3.9) consistant respectivement en : (i) l'évaluation de la similarité structurelle, (ii) l'évaluation des préférences de qualité et enfin (iii) l'évaluation de la similarité globale entre un modèle de processus cible et une requête à préférences.

Telle qu'illustrée dans la figure 3.10, la première procédure évalue les exigences structurelles de la requête et calcule la similarité structurelle entre chaque graphe de modèle de processus cible et le graphe requête, en considérant l'ensemble des opérations d'édition nécessaires pour transformer le graphe requête en le graphe cible. La seconde, quant à elle, détermine la satisfaction des préférences utilisateur sur les aspects de qualité de services (i.e. la fiabilité, le temps de réponse, le coût, ...). Elle évalue le degré de satisfaction des préférences obligatoires et souples. Enfin, la dernière procédure permet de calculer un degré global représentant la mesure dans laquelle un modèle de processus cible satisfait les exigences fonctionnelles et non-fonctionnelles de la requête.

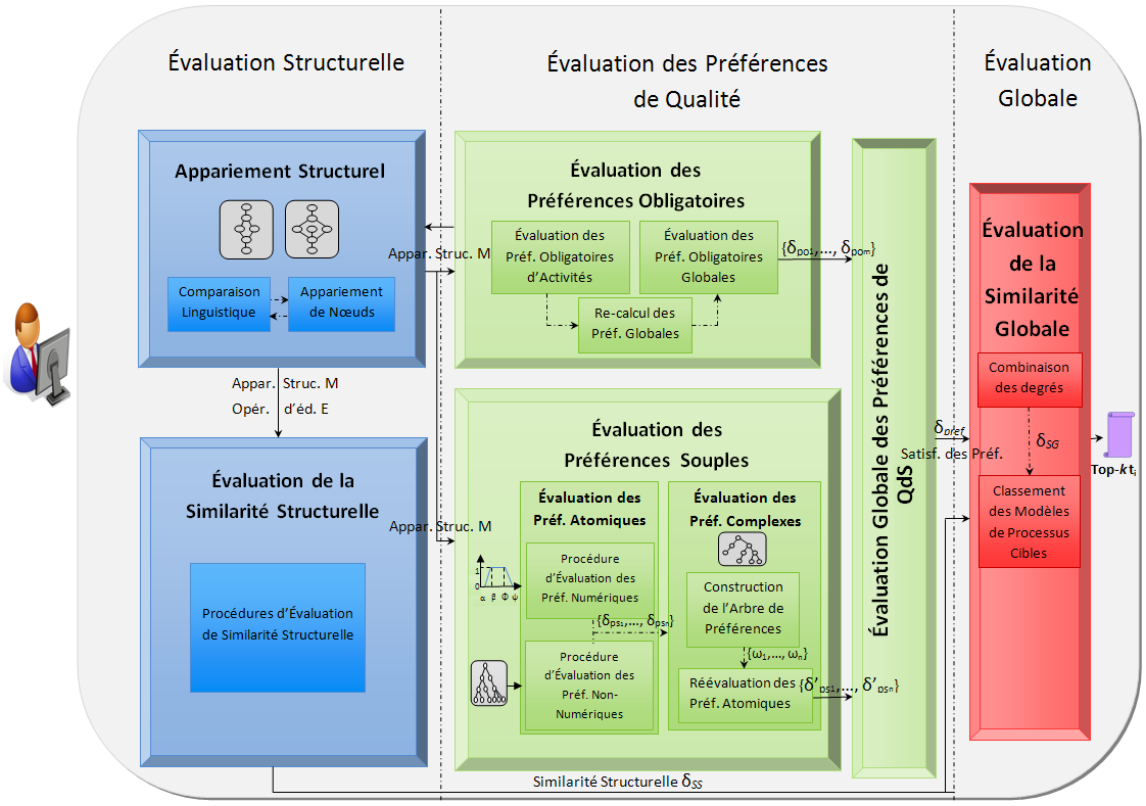



FIGURE 3.10 – Méthodologie de sélection de modèles de processus.

### 3.6.2 Méthodologie d'évaluation des modèles de processus

Pour répondre à un modèle de processus requête enrichi avec des préférences utilisateur sur les aspects de qualité de services, nous proposons une méthodologie d'évaluation permettant de prendre en compte la structure de chaque modèle de processus cible et ses informations de QdS. Cette méthodologie a plusieurs avantages, parmi lesquels nous citons : (i) la possibilité d'exprimer des préférences de qualité à différents niveaux de granularité (i.e., sur la globalité du modèle de processus et aussi au niveau des activités le composant), (ii) l'évaluation de ces préférences à l'aide de la théorie des ensembles flous pour mieux répondre aux besoins de chaque utilisateur et assurer les propriétés de subjectivité et de commensurabilité et enfin (iii) les réponses vides ou pléthoriques sont évitées en classant les modèles de processus réponses du plus satisfaisant au moins satisfaisant suivant à la fois les exigences fonctionnelles et non-fonctionnelles de l'utilisateur. La méthodologie proposée se décline en huit phases :

Considérons dans ce qui suit une requête à préférences $q$ et un modèle de processus cible $t$ représentés sous forme de graphes tel que discuté dans la section 3.2.

#### Phase 1 : L'appariement structurel

La première étape de la méthodologie d'évaluation consiste en l'appariement structurel entre les modèles de processus cibles et la requête utilisateur, pour déterminer les correspondances entre les

activités cibles et celles de la requête. En d'autres termes, pour l'évaluation du modèle de processus $t$ par rapport à la requête $q$, il est nécessaire d'identifier pour chaque activité de $q$, l'activité cible de $t$ qui lui correspond sémantiquement. Dans la figure 3.10, cet appariement est réalisé par le module « *Appariement structurel* ».

Le module « *Appariement structurel* » dans le système développé pour évaluer l'approche de sélection de modèles de processus proposée est basé sur l'algorithme proposé dans [59]. Ce dernier est une adaptation de l'algorithme « Error-Correcting Subgraph Isomorphism Detection (i.e., ECSI-Detection) » de Messmer et Bunke [104] pour l'appariement des modèles de processus. Il permet de retrouver l'appariement le moins coûteux parmi tous les appariements possibles entre les activités d'un modèle de processus cible et celles d'une requête utilisateur. Cet algorithme reçoit en entrée le graphe d'un modèle de processus cible et le graphe du modèle de processus requête, et renvoie par la suite l'appariement le moins coûteux, c'est-à-dire l'ensemble $M$ des correspondances entre activités et l'ensemble $E$ des opérations d'édition nécessaires pour transformer la requête en graphe de processus cible.

Un appariement entre $q$ et $t$ est un ensemble de paires $(v, w)$ où $v$ est une activité de $q$ et $w$ est une activité de $t$. Les opérations d'éditions considérées sont celles utilisées dans le domaine de graphes, à savoir : (i) la suppression de nœuds ; (ii) la suppression d'arêtes ; (iii) l'insertion de nœuds ; (iv) l'insertion d'arêtes ; (v) la substitution de nœuds et, (vi) la substitution d'arêtes.

Pour pouvoir choisir un meilleur appariement entre $q$ et $t$, l'algorithme de Gater *et al.* [59] calcule un coût pour chaque mise en correspondance entre les activités de $q$ et celles de $t$. Pour cela, les degrés de similarité structurelle entre les activités appariées et le coût des opérations d'édition nécessaires pour transformer $q$ en $t$, sont pris en considération.

Le degré de similarité structurelle entre une activité requête et une activité cible dépend de deux informations importantes : (i) leur position dans les graphes (i.e., les nœuds du flux de contrôle, les nœuds prédécesseurs et les nœuds successeurs) et (ii) leur similarité sémantique. Toutefois, deux activités appariées sont sémantiquement similaires si la distance entre leurs noms, leurs entrées et leurs sorties, est inférieure à un certain seuil défini par l'utilisateur.

La figure 3.4 page 82 montre un appariement structurel entre le graphe requête $q_1$ et le graphe cible du module de processus $t_1$. Cet appariement retourne la similarité structurelle, $ss(v, w)$, entre chaque paire d'activités appariées $(v, w)$, calculée en utilisant la mesure proposée dans [59].

**Phase 2 : L'évaluation des préférences obligatoires d'activités**

À partir d'un appariement obtenu par le module « *Appariement structurel* » entre les activités de la requête et celles d'un modèle de processus cible, les préférences obligatoires d'activités sont vérifiées. Cela permet d'éliminer les modèles de processus ne satisfaisant pas au moins une de ces préférences, et donc de renforcer le processus de filtrage de la première phase.

En d'autres termes, lorsque deux activités appariées $(v, w)$ sont considérées sémantiquement similaires, le module « *Évaluation des préférences obligatoires* » présenté dans la figure 3.10, vérifie si toutes les préférences obligatoires définies au niveau de l'activité requête $v$ sont satisfaites par les annotations correspondantes de l'activité cible $w$. De plus, durant cette vérification, le module

calcule le degré de satisfaction $\delta_{po_i}$ de chaque préférence obligatoire de $v$ par rapport à son annotation correspondante dans l'activité $w$. Cette phase retourne donc l'ensemble des degrés de satisfaction des préférences obligatoires d'activités de la requête utilisateur, noté $\{\delta_{po_1}, ..., \delta_{po_j}\}$.

**Phase 3 : Recalculer les annotations globales de qualité**

Suivant l'appariement structurel obtenu dans la première phase, les annotations de qualité globales du modèle de processus cible peuvent changer et donc doivent être recalculées en utilisant des fonctions d'agrégation [90]. Par exemple, la fiabilité globale d'un modèle de processus peut être égale à la valeur minimale des fiabilités des activités le composant. En effet, les nouvelles annotations recalculées seulement à partir des activités appariées avec celles de la requête, peuvent rendre le modèle de processus plus désirable suivant les préférences globales obligatoires de la requête.

Considérons, par exemple, l'appariement $M$ présenté dans la figure 3.4. La fiabilité du modèle de processus $t_1$ suivant cet appariement structurel est devenu 75% plutôt que 65%, ce qui rend $t_1$ plus au moins satisfaisant par rapport à la préférence obligatoire globale de la requête $q_1$.

Cette phase est réalisée par le module « *Évaluation des préférences obligatoires* » et retourne l'ensemble des valeurs des annotations globales du modèle de processus cible recalculées à partir des valeurs des annotations des activités appariées avec les activités de la requête.

**Phase 4 : L'évaluation des préférences obligatoires globales**

À partir des annotations globales obtenues dans la phase précédente, le module « *Évaluation des préférences obligatoires* » vérifie si toutes les préférences obligatoires globales sont satisfaites par les annotations globales correspondantes dans le modèle de processus cible. Dans le cas où au moins une seule préférence obligatoire globale n'est pas satisfaite, le modèle de processus cible est éliminé et supprimé de l'ensemble des réponses candidates à la requête.

Pour cela, le degré de satisfaction de chaque préférence obligatoire globale par rapport à son annotation globale correspondante, est calculé. Ainsi, le module « *Évaluation des préférences obligatoires* » retournent l'ensemble des degrés de satisfactions des préférences obligatoires globales, noté $\{\delta_{po_{j+1}}, ..., \delta_{po_m}\}$.

**Phase 5 : L'évaluation des préférences souples**

Cette phase consiste à évaluer les préférences souples définies par l'utilisateur, en se basant sur l'appariement structurel obtenu dans la phase (1). Pour cela, le module « *Évaluation des préférences souples* » de la méthodologie d'évaluation de modèles de processus proposée (voir la figure 3.10), calcule en premier lieu la satisfaction des préférences atomiques, évalue ensuite les préférences complexes pour enfin recalculer la satisfaction des préférences atomiques en prenant en considération leurs poids d'importance vis-à-vis de l'utilisateur.

En d'autres termes, pour chaque paire d'activités appariées $(v, w)$, le degré de satisfaction de chaque préférence atomique $\delta_{ps_i}$ par rapport à son annotation correspondante, est évalué tel que présenté dans la section 3.5. Pour chaque préférence atomique globale, son degré de satisfaction $\delta_{ps_k}$ est également calculé suivant son annotation globale correspondante.

Ensuite, les préférences complexes sont évaluées en construisant l'arbre de préférences $t_p$ correspondant, et les degrés de satisfaction des préférences atomiques précédemment évalués sont recalculés en prenant en compte leurs poids d'importance par rapport à l'utilisateur (voir la section 3.5). Enfin, l'ensemble des degrés de satisfaction des préférences souples de la requête, noté $S_{ps} = \{\delta_{ps_1}, ..., \delta_{ps_n}\}$, est retourné par le module « *Évaluation des préférences souples* ».

**Phase 6 : L'évaluation globale des préférences**

Afin d'évaluer la satisfaction globale des préférences de l'utilisateur sur les aspects de qualité des modèles de processus cibles, le module « *Évaluation globale des préférences* » procède à la combinaison des degrés de satisfaction des préférences obligatoires et souples des ensembles $S_{po} = \{\delta_{po_1}, ... , \delta_{po_m}\}$ et $S_{ps} = \{\delta_{ps_1}, ..., \delta_{ps_n}\}$, obtenus respectivement par les modules « *Évaluation des préférences obligatoires* » et « *Évaluation des préférences souples* ». Cette combinaison nous permet d'évaluer la mesure, notée $\delta_{pref}$, dans laquelle un modèle de processus cible satisfait les préférences obligatoires et les préférences souples de l'utilisateur.

Il est important de noter que la méthode calculant la satisfaction globale des préférences obligatoires, notée $\delta_{S_{po}}$, et celle des préférences souples, notée $\delta_{S_{ps}}$, est également implémentée dans le module « *Évaluation globale des préférences* » sans avoir recours à leur agrégation.

Pour assurer encore plus la propriété de subjectivité, les mesures d'agrégation présentées dans la sous-section 3.5.3 ont été implémentées dans le module « *Évaluation globale des préférences* » de la figure 3.10. Ceci permet donc à l'utilisateur de choisir les mesures qui lui conviennent le plus suivant ses objectifs et son point de vue.

**Phase 7 : L'évaluation de la similarité structurelle**

Dans cette phase, le module « *Évaluation de la similarité structurelle* » illustré dans la figure 3.10 mesure la similarité structurelle, notée $\delta_{SS}$, entre le graphe de la requête $q$ et celui du modèle de processus cible $t$. Pour cela, il reçoit en entrée les exigences structurelles de l'utilisateur et les résultats obtenus dans la première phase, à savoir l'appariement $M$ et l'ensemble des opérations d'édition nécessaires pour transformer le graphe requête en le graphe cible.

À partir de ces informations, le degré de similarité structurelle entre $q$ et $t$ est calculé en appliquant une des mesures d'évaluation définies dans la sous-section 3.5.4.

**Phase 8 : L'évaluation globale des modèles de processus**

Avant de retourner l'ensemble des modèles de processus cibles réponses à la requête, il est important de les ordonner des plus intéressants aux moins intéressants vis-à-vis de l'utilisateur. Pour cela, le module « *Évaluation globale des modèles de processus* » implémente un ensemble de mesures permettant le classement de ces réponses. Dans la sous-section 3.5.5, quatre mesures sont proposées.

Ces mesures sont basées soit sur l'agrégation des deux degrés $\delta_{pref}$ et $\delta_{SS}$ pour obtenir un seul degré global, noté $\delta[q, t]$, ou bien sur la non-agrégation des deux degrés. Une fois de plus, la mesure à appliquer est au choix de l'utilisateur.

## 3.7 Conclusion

Ce travail a fait l'objet d'une collaboration entre l'équipe PRiSM de l'Université de Versailles Saint-Quentin-en-Yvelines et l'équipe PILGRIM de l'IRISA dans le cadre du projet ANR-AOC. Nous avons d'abord rappelé les principes du système de découverte et de sélection de modèles de processus proposé au sein de PRiSM. Puis, nous avons présenté dans ce chapitre une méthode basée sur les techniques de la théorie des ensembles flous pour évaluer des modèles de processus. Cette méthode prend en considération les contraintes fonctionnelles des utilisateurs et leurs exigences non-fonctionnelles sur des aspects de QdS. Nous avons décrit les procédures et les algorithmes d'évaluation des préférences de qualité, à savoir, les préférences obligatoires et les préférences souples. Nous avons également présenté les mesures calculant la similarité structurelle, la satisfaction des préférences de qualité et leur agrégation. Chacune de ces mesures fournit une sémantique claire et simple à interpréter par les utilisateurs.

Par la suite, nous avons décrit l'architecture de la méthodologie de sélection de modèles de processus en définissant ses différents composants et modules et détaillons chacune de ses phases. Cette méthodologie considère la similarité structurelle et la satisfaction des préférences de qualité et détermine dans quelle étape ces préférences sont prises en compte durant le processus d'évaluation de modèles de processus. Elle spécifie également les mesures d'agrégation nécessaires pour évaluer la similarité structurelle et la satisfaction des préférences.

La méthodologie proposée présente plusieurs avantages :

- Elle prend en considération la similarité structurelle et la satisfaction des préférences de qualité.
- Elle renforce le processus de filtrage de l'algorithme d'appariement structurel en sélectionnant non seulement les modèles de processus structurellement similaire mais aussi satisfaisant les préférences obligatoires de l'utilisateur.
- Elle offre plusieurs mesures d'évaluation de la similarité structurelle, de la satisfaction des préférences et de la similarité globale entre deux modèles de processus.
- Elle offre plus de flexibilité. En effet, elle permet à l'utilisateur de combiner de différentes façons ces mesures pour personnaliser l'appariement de modèles de processus.

Nous présentons dans le chapitre suivant le prototype développé pour implémenter les mesures et la méthodologie proposées. Ensuite, nous montrons les résultats des différentes expérimentations que nous avons menées et qui nous ont permis d'évaluer et de valider la méthodologie proposée.

# Chapitre 4

# Implémentation et expérimentation

## 4.1 Introduction

Dans ce chapitre, nous présentons le prototype de sélection de modèles de processus et les expérimentations conduites pour valider l'approche proposée et évaluer la pertinence de ses résultats [8, 9, 6]. Ce prototype représente donc une mise en œuvre de la méthode d'évaluation de modèles de processus présentée dans le chapitre 3.

Cette mise en œuvre est rendue possible grâce à l'outil « *MatchMaker* » développé dans le cadre du projet ANR-AOC par l'équipe PRiSM [92]. Notre apport dans ce cadre a porté sur la spécification et la définition des métriques utilisées pour l'élaboration des tests et la réalisation des expérimentations.

En sections 4.2 et 4.3, nous décrivons le prototype développé pour intégrer les préférences sur les aspects de QdS dans l'évaluation de modèles de processus. Nous présentons également dans la section 4.4, l'analyse de la complexité des procédures implémentées, à savoir, les procédures d'évaluation des préférences obligatoires et des préférences souples et celles des mesures présentées dans le chapitre 3. La section 4.5 décrit l'environnement de tests considéré ainsi que l'ensemble des résultats expérimentaux obtenus à partir du prototype développé par l'équipe PRiSM. Enfin, la section 4.6 conclut le chapitre.

## 4.2 Architecture du prototype

L'objectif étant de sélectionner les modèles de processus les plus similaires à la requête de l'utilisateur et satisfaisant ses préférences de QdS, l'outil dit « *matchMaker* » a été développé et implémenté en se basant sur la méthodologie d'évaluation présentée dans le chapitre 3 en section 3.6. Cet outil est utilisé pour démontrer la pertinence de la méthode d'évaluation de modèles de processus proposée.

Comme illustré dans la figure 4.1, l'architecture de « *matchMaker* » est composé des modules suivants :

– **Le dépôt ou référentiel de modèles de processus :** ce référentiel représente la base dans

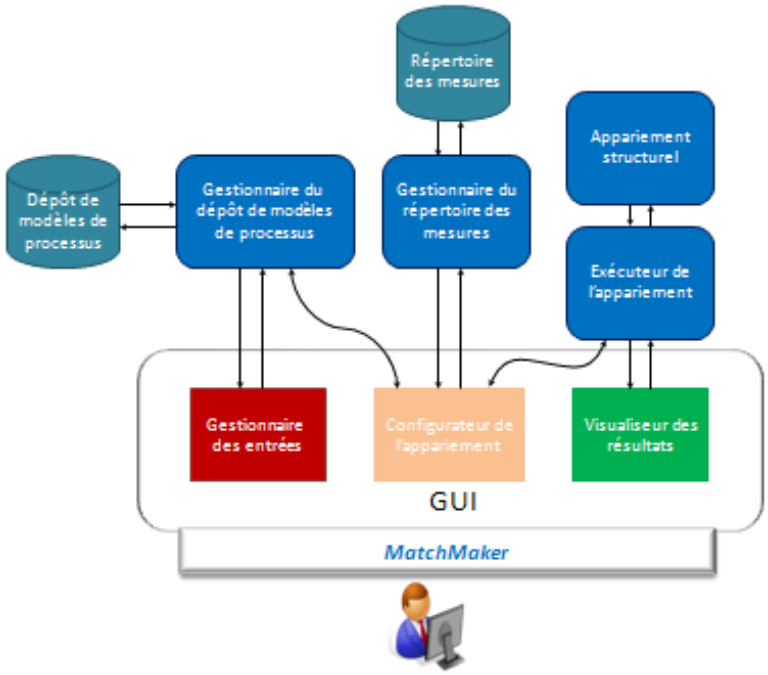



FIGURE 4.1 – Architecture du prototype de sélection de modèles de processus.

laquelle l'utilisateur peut stocker les documents XML des requêtes et des modèles de processus cibles.

- **Le gestionnaire du dépôt de modèles de processus :** Ce module a pour rôle l'exécution des opérations de base pour gérer le dépôt de modèles de processus, à savoir l'accès à la base et le stockage de modèles de processus.
- **Le répertoire des mesures :** ce répertoire stocke l'ensemble des mesures calculant la similarité structurelle, la satisfaction des préférences et la similarité globale entre une requête et chaque modèle de processus cible, tel que présenté dans le chapitre précédent.
- **Le gestionnaire du répertoire de mesures :** ce gestionnaire se charge de l'accès aux mesures citées ci-dessus ainsi que de leur exécution.
- **Le module « Appariement structurel » :** la mise en correspondance structurelle entre une requête utilisateur et un modèle de processus cible est réalisée au moyen du module **« Appariement Structurel »**.
- **Exécuteur de l'appariement** : ce module gère l'exécution de l'algorithme d'appariement structurel en utilisant les paramètres définis par l'utilisateur via le module « **Configurateur d'appariement** ».
- **Le module GUI (Graphic User Interface) :** ce module représente l'interface graphique permettant à l'utilisateur de spécifier sa requête et l'ensemble des modèles de processus cibles à évaluer, pour retourner par la suite la similarité structurelle, la satisfaction des préférences et la similarité globale entre la requête et chaque modèle de processus cible. Plus exactement, ce module permet à l'utilisateur : (i) de gérer le dépôt des modèles de processus cibles ; (ii)

de définir ses requêtes pour interroger le référentiel de graphes cibles ; (iii) de configurer les paramètres d'initialisation et lancer l'exécution de la procédure d'appariement ; et enfin (iv) de visualiser et manipuler les résultats de sa recherche. Pour cela, le module GUI se constitue de trois sous-modules :

- **Gestionnaire des entrées :** ce sous-module fournit à l'utilisateur les opérations de base pour gérer le dépôt de modèles de processus et lui permet de visualiser les modèles de processus requêtes et cibles.
- **Configurateur d'appariement :** pour personnaliser la recherche de modèles de processus, le configurateur d'appariement permet à l'utilisateur de définir et d'initialiser les paramètres intervenant dans le calcul de la similarité structurelle, la satisfaction des préférences et la similarité globale et d'autres paramètres tels que le coût des opérations d'édition et l'ontologie des attributs non-numériques.
- **Visualiseur des résultats :** comme son nom l'indique, le visualiseur des résultats fournit à l'utilisateur la liste des modèles de processus réponses à la requête. Il donne également la possibilité à l'utilisateur d'ordonner les réponses suivant soit leur degré de similarité structurelle, leur degré de satisfaction des préférences ou bien leur degré de similarité globale. Enfin, l'utilisateur peut aussi visualiser l'appariement structurel entre chaque modèle de processus réponse et le graphe de sa requête.

Dans le cadre du projet AOC, une version Web offrant les fonctionnalités de base de du prototype développé par PRiSM, appelé « *Web matchMaker* », est implémenté et accessible sur le site http ://aoc.irit.fr/.

## 4.3 Description du prototype

L'outil « *matchMaker* » implémenté représente le support nous permettant d'évaluer et de valider l'approche de sélection de modèles de processus décrite en section 3.5 du chapitre 3. Comme présenté précédemment, il se caractérise d'une interface permettant à l'utilisateur de spécifier sa requête et l'ensemble des modèles de processus cibles à évaluer, pour retourner par la suite la similarité structurelle, la satisfaction des préférences et la similarité globale entre la requête et chaque modèle de processus cible. Ceci est réalisé en exécutant une procédure d'appariement entre modèles de processus.

La procédure d'appariement structurel de « *matchMaker* » implémente l'algorithme ECSI-Detection adapté aux modèles de processus pour prendre en considération leurs noms, leurs entrées et leurs sorties [59]. Cette algorithme est très basique et permet d'effectuer un appariement un-à-un et non pas un-à-plusieurs, i.e., un appariement structurel entre la requête de l'utilisateur et un et un seul modèle de processus cible. Néanmoins, le « *matchMaker* » peut être utilisé comme plateforme pour la recherche de modèles de processus par toute autre application traitant de grands volumes d'information, en l'améliorant avec des techniques d'indexation et d'autres algorithmes d'appariement plus efficaces.

Le « *matchMaker* » est très simple et facile à utiliser. La première étape à effectuer par l'utilisateur consiste à choisir et à télécharger sa base de modèles de processus ainsi que son modèle de processus requête. De cette façon, l'utilisateur peut visualiser et analyser la structure de graphe ainsi que l'ensemble des annotations ou de préférences de qualité de chaque modèle de processus cible ou requête en une simple sélection. Dans la deuxième étape, l'utilisateur procède à l'initialisation de certains paramètres nécessaires pour lancer le processus de recherche de modèles de processus. Ceci a pour but de personnaliser la recherche en spécifiant :

- les paramètres de l'algorithme d'appariement structurel tels que le coût de chaque opération d'édition, les seuils de sélection, ... ;
- une mesure de calcul de satisfaction globale des préférences parmi celles proposées dans la sous-section 3.5.3 ;
- une mesure de calcul de similarité structurelle parmi celles présentées dans la sous-section 3.5.4 et
- une mesure calculant la similarité globale entre chaque modèle de processus cible et la requête, parmi les mesures définies dans la sous-section 3.5.5.

Une fois que tous les paramètres sont définis, l'utilisateur peut lancer le processus d'appariement à l'aide du bouton « *Execute* ». Ceci permet de retourner en troisième étape, l'ensemble des modèles de processus réponses à la requête avec le temps d'exécution de chaque procédure d'évaluation ainsi que leurs degrés de similarité structurelle, de satisfaction globale des préférences de qualité et de similarité globale. Enfin, l'utilisateur peut visualiser dans la partie « *Mapping Viewer* », l'appariement effectué entre la requête et chaque modèle de processus réponse en cliquant sur le bouton « Mapping ».

## 4.4 Analyse de la complexité

Le temps d'exécution de l'approche de recherche de modèles de processus proposée dépend du temps nécessaire pour l'exécution de ses trois procédures principales, à savoir,

- la procédure d'évaluation des préférences obligatoires ;
- la procédure d'évaluation des préférences souples et
- les mesures de calcul de similarité structurelle, de satisfaction globale des préférences et de similarité globale.

En conséquence, l'analyse de la complexité temporelle des algorithmes peut être réalisée en trois étapes (voir [92]). Pour ce faire, nous considérons dans ce qui suit, qu'une requête utilisateur possède $n$ nœuds d'activités et qu'un modèle de processus cible, quant à lui, possède $m \geq n$ nœuds d'activités, où chaque activité est annotée avec $k$ différents attributs de qualité.

**La complexité de la procédure d'évaluation des préférences obligatoires.** L'analyse de la complexité de la procédure d'évaluation des préférences obligatoires peut être effectuée de deux façons différentes : (i) une analyse séparée de celle de la complexité du processus d'appariement structurel lorsque ce dernier est exécuté avant la procédure d'évaluation des préférences obligatoires et

(ii) une analyse entrelacée lorsque la procédure d'évaluation des préférences obligatoires est exécutée durant l'appariement structurel tel que présenté en section 3.6 page 102 du chapitre précédent.

Supposons dans ce qui suit, que chaque activité requête est annotée avec $k$ préférences obligatoires de différents attributs de qualité.

**Cas (i).** Considérons que chaque activité requête est appariée avec une activité du graphe cible. Ainsi, l'appariement entre la requête et le modèle de processus cible est composé de $n$ correspondances entre nœuds d'activités. Le pire cas de cette situation est lorsque toutes les préférences obligatoires sont satisfaites par la cible, car il suffit qu'une seule préférence obligatoire ne soit pas vérifiée pour que la cible soit éliminé de l'espace de recherche. Ainsi, le temps nécessaire pour trouver l'annotation correspondante à une préférence obligatoire donnée dans chaque activité requête est $O(k)$. Le temps nécessaire pour évaluer toutes les préférences obligatoires d'une activité requête est donc $O(k^2)$. Par conséquent, l'évaluation de toutes les préférences obligatoires des $n$ activités de la requête nécessite un temps égal à $O(n.k^2)$. Autrement dit, le temps nécessaire pour évaluer toutes les préférences obligatoires de l'utilisateur est $O(r.k)$ où $r = n.k$ dénote le nombre de préférences obligatoires de la requête. Ainsi, la complexité de la procédure d'évaluation des préférences obligatoires exécutée après l'appariement structurel est *polynomiale*. Enfin, le temps global pour l'exécution de la procédure d'évaluation des préférences obligatoires et de l'appariement structurel est $O(n.m^n + r.k)$.

**Cas (ii).** Le pire cas de l'exécution de la procédure d'évaluation des préférences obligatoires durant l'appariement structurel est lorsque (i) chaque activité requête est sémantiquement similaire avec toute autre activité cible et (ii) toutes les préférences obligatoires de chaque activité requête sont satisfaites par les annotations de toute autre activité cible. Dans ce cas, la complexité de la procédure d'évaluation des préférences obligatoires est *polynomiale* et le temps global pour son exécution et celle de l'algorithme d'appariement structurel est $O(n.m^n.k^2)$.

En considérant que les activités requêtes ont des valeurs de QdS différentes, la procédure d'évaluation des préférences obligatoires durant l'appariement structurel permet d'éviter plusieurs appariements possibles entre la requête et un modèle de processus cible, lorsqu'au moins une préférence obligatoire n'est pas satisfaite. Ceci permet donc de réduire la taille de l'arbre de recherche [90] construit durant l'appariement structurel.

Cependant, la supposition « toutes les activités cibles satisfont toutes les préférences obligatoires de chaque activités requêtes » considère que toutes ces activités ont des valeurs similaires des attributs de qualité, ce qui n'est pas généralement le cas dans les bases de données de modèles de processus.

**La complexité de la procédure d'évaluation des préférences souples.** Les préférences souples comme décrit en section 3.4 page 76 sont soit atomiques ou bien complexes en utilisant les relations de préférences de priorité ou de Pareto (voir la définition 3.4.3). Ainsi, le temps global pour évaluer les préférences souples est composé (i) du temps nécessaire pour évaluer les préférences souples atomiques et (ii) le temps nécessaire pour construire l'arbre de préférences et ré-évaluer les préférences atomiques en prenant en considération leurs poids d'importance suivant les préfé-

rences complexes. Ainsi, le pire cas dans l'évaluation des préférences souples est lorsque toutes les préférences atomiques font parties de la définition des préférences complexes.

Dans ce cas, supposons que chaque activité requête dispose de $k$ préférences souples atomiques de différents attributs de QdS et que toutes les relations de préférences (de priorité et de Pareto) sont définies (i.e. $O(k^2)$ préférences souples complexes).

Le temps nécessaire pour trouver l'annotation correspondante à chaque préférence atomique dans chaque activité requête est $O(k)$. Ainsi, le temps pour évaluer toutes les préférences atomiques d'une activité requête est $O(k^2)$. De plus, le temps d'évaluation d'une préférence atomique est soit constant dans le cas d'une préférence numérique, ou bien égal à $O(h)$ dans le cas d'une préférence non-numérique tel que $h$ est la hauteur de l'ontologie correspondante.

En considérant le cas où toutes les préférences atomiques sont non-numériques, l'évaluation des préférences atomiques des $n$ activités de la requête est effectuée en un temps égal à $O(n.k^2.h) = O(r.k.h)$, où $r = n.k$ représente le nombre de préférences souples définies dans la requête de l'utilisateur. Quant à la construction de l'arbre des préférences, elle nécessite un temps égal à $O(k^2)$ pour appliquer les règles de priorité exprimées dans les préférences complexes de la requête. Ainsi, la complexité temporelle de l'évaluation des préférences souples d'une requête utilisateur est égal à $O(n.k^2.h + k^2) \backsimeq O(n.k^2.h)$.

Enfin, la complexité des mesures d'évaluation des préférences souples est soit *polynomiale* dépendant du nombre d'activités requêtes et du nombre de préférences (la valeur de la hauteur $h$ des ontologies utilisées pour les préférences non-numériques est généralement petite) ou bien *constante* dans le cas où les préférences souples de la requête sont toutes des préférences numériques.

**La complexité de la procédure d'évaluation de la similarité structurelle.** Pour évaluer la similarité structurelle, le nombre d'activités requêtes appariées et le nombre d'opérations d'édition nécessaires pour transformer la requête en graphe cible, sont pris en considération dans les mesures proposées dans la section 3.5.4. Ainsi, le pire cas du calcul de la similarité structurelle est lorsque le nombre d'opérations d'édition est maximal. Pour cela, supposons que chaque activité requête (parmi les $n$ activités) est appariée avec une activité cible (parmi les $m$ activités du modèle de processus cible).

Ainsi, le coût d'opérations d'édition nécessaires pour transformer la requête en modèle de processus cible est maximal, lorsque toutes les arêtes ainsi que tous les nœuds de flux de contrôle de la requête sont à supprimer pour insérer par la suite ceux du modèle de processus cible (i.e. aucune arête et aucun nœud de flux de contrôle de la requête ne correspondent aux arêtes et aux nœuds de flux de contrôle du modèle de processus cible). En utilisant le système proposé dans [57], le nombre moyen des nœuds de flux de contrôle d'un modèle de processus est égal à la moitié de son nombre de nœuds d'activités.

En conséquence, la suppression de tous les nœuds de flux de contrôle de la requête et l'insertion de ceux de la cible, nécessite $n/2 + m/2$ opérations. De plus, le maximum d'arêtes qu'un modèle de processus peut avoir est $n^2$. Pour supprimer toutes les arêtes de la requête et insérer celles de la cible, il faut au maximum $n^2 + m^2$ opérations. Ainsi, le nombre d'opérations d'édition nécessaire

pour transformer la requête en le modèle de processus cible est au maximum $(n^2+n/2+m^2+m/2)$. Enfin, la complexité de la procédure d'évaluation de la similarité structurelle est *polynomiale* et est égale à $O(n^2+m^2)$.

**La complexité de la procédure d'évaluation de la satisfaction globale des préférences.** Le temps nécessaire pour évaluer la satisfaction globale des préférences de l'utilisateur dépend du nombre des degrés de satisfaction des préférences de qualité. Ainsi, la complexité de l'évaluation de la satisfaction globale des préférences est *linéaire* et égale à $O(r')$, où $r'$ est le nombre des préférences de la requête utilisateur.

**La complexité de la procédure d'évaluation de la similarité globale.** La complexité des mesures de calcul de similarité globale entre deux modèles de processus proposées dans la section 3.5.5, est *constante* car elles ne dépendent que des degrés de la similarité structurelle et de la satisfaction globale.

En conclusion, la complexité de la méthode de sélection de modèles de processus proposée est *polynomiale*, qui est insignifiante et négligeable devant la complexité exponentielle de l'algorithme d'appariement structurel (i.e., $O(n.m^n)$).

## 4.5 Expérimentations

Dans cette section, nous présentons les expérimentations menées pour évaluer la performance et la fiabilité de la méthode de sélection de modèles de processus proposée. Ces expérimentations sont réalisées dans le cadre du projet ANR-AOC en collaboration avec l'équipe PRiSM de l'université de Versailles Saint-Quentin-en-Yvelines et ont pour objectif de :

- mesurer le temps nécessaire pour évaluer les préférences obligatoires durant l'appariement structurel ;
- évaluer la performance de la procédure calculant la satisfaction globale des préférences ;
- mesurer l'impact de l'intervention de l'utilisateur dans l'évaluation des préférences, i.e. évaluer les préférences avec et sans la prise en considération du point de vue de l'utilisateur ;
- évaluer la pertinence des résultats de l'approche proposée en appliquant les mesures proposées dans les sections 3.5.3, 3.5.4 et 3.5.5 et
- évaluer la fiabilité de l'approche comparée à celle proposée dans [90].

### 4.5.1 Environnement expérimental

Afin de montrer la faisabilité et la pertinence de l'approche de sélection de modèles de processus proposée, nous avons utilisé la collection présentée dans [57] qui contient 623 modèles de processus. Ces derniers ont une taille moyenne de 20 activités avec une taille minimale de 2 activités et une taille maximale composée de 83 activités.

Les valeurs des attributs de QdS de ces modèles de processus sont extraites de la base de données

QWS [10], qui est composée de 5000 services Web annotés avec les 7 attributs de qualité suivants : temps de réponse (ms), disponibilité (%), fiabilité (%), débit « throughput » (invocations/second), taux de réponses « successability » (%), compliance (%) et latence (ms). Nous avons également ajouté l'attribut de QdS, sécurité, dont les valeurs sont prises de l'ontologie proposée par Herzog *et al.* [72].

### 4.5.2 Résultats expérimentaux

Dans ce qui suit, nous présentons les expérimentations réalisées pour valider la méthode de sélection de modèles de processus proposée ainsi que les résultats obtenus en évaluant :

- la performance de la procédure d'évaluation des préférences obligatoires ;
- la performance de la procédure d'évaluation des préférences souples ; et
- la pertinence des mesures de similarité structurelle, de satisfaction globale des préférences et de similarité globale proposées dans le chapitre 3.

Dans chaque expérimentation, nous décrivons la base de modèles de processus utilisée ainsi que la méthode appliquée et les résultats obtenus.

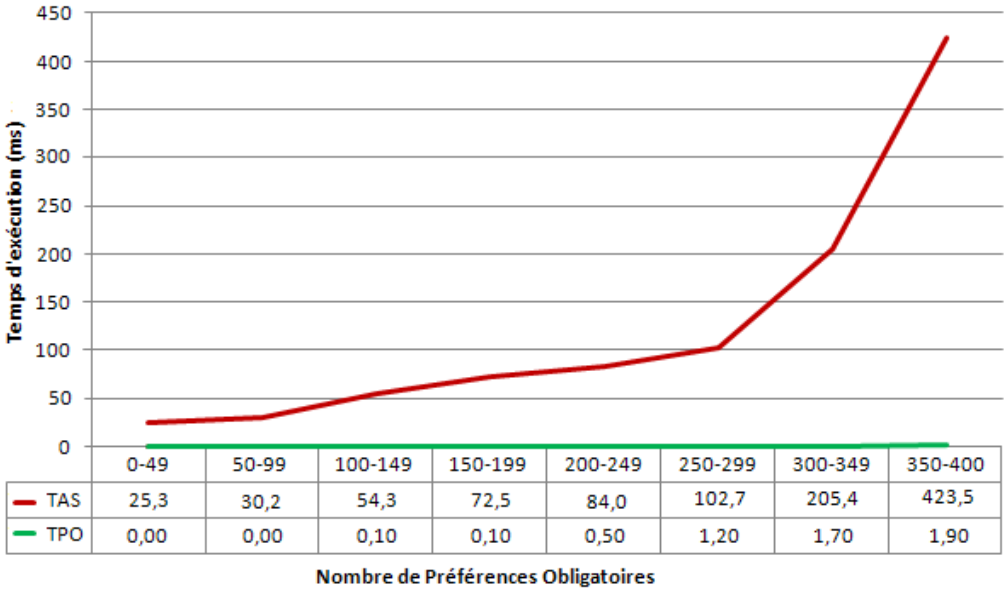
**1. Mesurer la performance de la procédure d'évaluation des préférences obligatoires**

Telle que présentée en section 4.4, la procédure d'évaluation des préférences obligatoires est dépendante du nombre de préférences obligatoires exprimées dans la requête de l'utilisateur. Par conséquent, il convient d'évaluer sa performance en calculant son temps d'exécution lorsque le nombre des préférences obligatoires varie.

Pour ce faire, une sous-collection de 150 modèles de processus a été sélectionnée au hasard à partir de la collection présenté dans [57]. De cette sous-collection, un ensemble de 40 modèles de processus a été construit aléatoirement pour représenter la collection de modèles de processus requêtes. Ces derniers ainsi que leurs activités ont été annotés avec le même ensemble de 8 préférences obligatoires de QdS (une préférence pour chacun des attributs de qualité cités ci-dessus, i.e. : disponibilité, fiabilité, débit, taux de réponses, compliance, latence et sécurité).

Quant aux 150 modèles de processus de la sous-collection de base, ils ont été utilisés comme modèles de processus cibles à évaluer suivant chaque requête. Pour cela, ces modèles de processus cibles ainsi que leurs activités ont été annotés avec les mêmes 8 annotations. Afin de prendre en considération toutes les mises en correspondance possibles entre une requête et un modèle de processus cible, les valeurs des annotations ont été choisies de telle sorte que toutes les préférences obligatoires soient satisfaites.

À partir des deux collections de modèles de processus requêtes et cibles, l'algorithme d'appariement structurel a été exécuté plusieurs fois entre chaque requête et les modèles de processus cibles, en variant le nombre de préférences obligatoires. Ainsi, le temps pris par la procédure d'évaluation des préférences obligatoires à chaque exécution a été mesuré et regroupé par des intervalles de nombre de préférences obligatoires évaluées.




| | 0-49 | 50-99 | 100-149 | 150-199 | 200-249 | 250-299 | 300-349 | 350-400 |
|---|---|---|---|---|---|---|---|---|
| TAS | 25,3 | 30,2 | 54,3 | 72,5 | 84,0 | 102,7 | 205,4 | 423,5 |
| TPO | 0,00 | 0,00 | 0,10 | 0,10 | 0,50 | 1,20 | 1,70 | 1,90 |

FIGURE 4.2 – Le temps moyen de l'appariement structurel (TAS) et le temps moyen de l'évaluation des préférences obligatoires (TPO) en fonction du nombre de préférences obligatoires.

La figure 4.2 présente le temps moyen de l'appariement structurel (TAS) et le temps moyen de l'évaluation des préférences obligatoires (TPO) en variant le nombre de préférences obligatoires. Les requêtes ayant plus de 50 activités, donc plus de 400 préférences obligatoires, ont été omises car leur évaluation était irréalisable.

Néanmoins, comme le montre la figure 4.2, le temps supplémentaire pris pour évaluer les préférences obligatoires est *insignifiant* et *négligeable* comparé au temps pris par le processus d'appariement structurel. En effet, il représente moins de 1% du temps global.

**2. Mesurer la performance de la procédure d'évaluation des préférences souples**

Similairement aux préférences obligatoires, la procédure d'évaluation des préférences souples dépend du nombre de préférences souples exprimées dans la requête de l'utilisateur. Pour cela, l'évaluation de sa performance a été réalisée en mesurant son temps d'exécution lorsque le nombre des préférences souples varie.

Pour cela, une base de modèles de processus a été construite en choisissant au hasard 150 modèles de processus à partir de la collection de [57]. Par la suite, 40 modèles de processus ont été sélectionnés aléatoirement à partir de la base pour former une collection de requêtes. Chacun de ces modèles de processus ainsi que ses activités ont été annotés avec 8 différentes préférences souples atomiques et 8 préférences souples complexes (i.e., des préférences de priorité et de Pareto, voir la définition 3.4.3 page 8), générées d'une manière aléatoire.

La collection des modèles de processus cibles, quant à elle, a été composée des 150 modèles de processus de la base d'origine. Chaque modèle de cette collection ainsi que l'ensemble de ses activités ont été annotés avec les 8 annotations de QdS dont les valeurs ont été sélectionnées aléatoirement à partir de la base QWS de [10] et de l'ontologie de sécurité de [72].

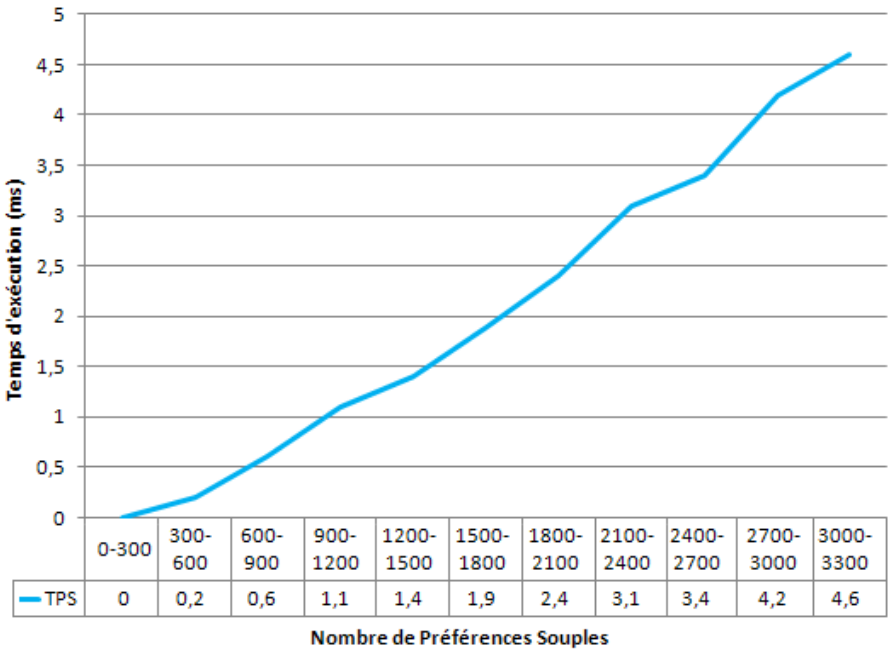



FIGURE 4.3 – Le temps moyen de l'évaluation des préférences souples (TPS) en fonction du nombre de préférences souples.

À partir de ces deux collections, l'appariement structurel a été d'abord réalisé manuellement entre chaque requête et les modèles de processus cibles. Ensuite, le temps pris pour évaluer les préférences souples a été mesuré pour chaque paire de modèles de processus appariés. Ces temps d'exécution ont été regroupés dans des intervalles représentant le nombre de préférences souples évaluées.

Le temps moyen de l'évaluation des préférences souples (TPS) est illustré dans la figure 4.3. Les résultats obtenus dans cette expérimentation ont montré que le temps nécessaire pour évaluer les préférences souples est *négligeable* comparé au temps pris par le processus d'appariement structurel. De plus, l'évaluation des préférences souples nécessite moins de temps que l'évaluation des préférences obligatoires. En effet, dans la procédure d'évaluation des préférences obligatoires, une même préférence peut être évaluée autant de fois que son activité est mise en correspondance avec une nouvelle activité, tandis que dans l'évaluation des préférences souples, chaque préférence est évaluée une et une seule fois car l'appariement structurel est déjà établi.

**3. Évaluer la pertinence des mesures de similarité structurelle, de satisfaction des préférences et de similarité globale.**

Leur objectif principal étant de fournir des résultats ayant une sémantique claire et simple, les mesures proposées dans la section 3.5 doivent être également efficaces et pertinentes en retournant les modèles de processus les plus intéressants vis-à-vis de l'utilisateur. Pour cela, des expérimentations ont été réalisées pour évaluer l'efficacité de ces mesures comparées aux mesures fournissant des résultats sans sémantique explicite, c'est-à-dire : (i) mesurer la pertinence des résultats en utilisant l'indice NDCG (Normalized Discounted Cumulative Gain), qui est une mesure évaluant l'efficacité

d'un ordonnancement en se basant sur la position d'une réponse dans la liste des résultats et (ii) comparer les résultats de nos mesures avec ceux des mesures n'ayant aucune sémantique. En d'autres termes, l'objectif de ces expérimentations est de déterminer les mesures qui ordonnancent le mieux les modèles de processus réponses en considérant les exigences structurelles et de qualité de l'utilisateur.

Pour ce faire, nous avons construit une base de modèles de processus en sélectionnant au hasard 65 modèles de processus à partir de la collection présentée dans [57]. À partir de cette base, un ensemble de 5 modèles de processus a été choisi aléatoirement pour construire la collection de requêtes. Chaque modèle de processus requête ainsi que certaines de ses activités ont été d'abord annotés avec des préférences textuelles décrivant des préférences atomiques et complexes en langage naturel. Ensuite, le graphe annoté correspondant à chaque requête de modèle de processus a été généré en utilisant le modèle présenté dans la section 3.4. Quant à la collection des cibles, elle a été composée des 65 modèles de processus de la base d'origine où chaque modèle ainsi que ses activités ont été annotés avec 8 attributs de QdS. Les valeurs de ses attributs ont été sélectionnées aléatoirement de la base QWS de [10] et de l'ontologie de sécurité de [72].

Afin d'évaluer l'efficacité des mesures proposées dans cette thèse, il est important de définir pour chaque requête l'ordonnancement idéal de ses modèles de processus réponses correspondants. Ceci permet de comparer l'ordonnancement résultant de chaque mesure avec l'ordonnancement idéal de chaque requête, dans l'objectif de déterminer la mesure qui correspond le mieux aux attentes de l'utilisateur.

*Définition de l'ordonnancement idéal pour chaque requête.* Un groupe d'experts a été invité pour analyser manuellement la satisfaction de chaque modèle de processus cible par rapport aux exigences structurelles et de qualité exprimées dans les requêtes textuelles. Ensuite, ils ont donné trois notes de 1 à 7 (1 pour très différents, 7 pour fortement similaires) correspondant respectivement au classement suivant (i) la similarité structurelle, (ii) la satisfaction globale des préférences de qualité et (iii) la similarité globale en considérant à la fois la similarité structurelle et la satisfaction des préférences. Enfin, les top-k meilleurs modèles de processus de chaque requête ont été identifiés et ordonnés par les experts. Il convient de noter que le classement basé sur la satisfaction des préférences a été réalisé sur la base des top-k modèles de processus structurellement similaires.

*Ordonnancement résultant de chaque mesure évaluée.* Pour chaque requête et un modèle de processus cible, un appariement structurelle a été d'abord déterminé manuellement. Ensuite, les mesures d'évaluation des modèles de processus cibles ont été exécutées pour déterminer les ordonnancements correspondant à chaque requête. Enfin, à partir de chaque ordonnancement les top-k modèles de processus ont été sélectionnés et leurs degrés NDCG ont été calculés et évalués.

La figure 4.4 présente les degrés NDCG obtenus à partir des différentes mesures implémentés pour évaluer : (i) la similarité structurelle, (ii) la satisfaction des préférences et (iii) la similarité globale entre chaque requête utilisateur et un modèle de processus cible de la base. Plus le degré NDCG est grand, plus l'ordonnancement proposé par la mesure correspondante est proche de l'ordonnancement idéal défini par les experts.

Dans la catégorie, *Similarité Structurelle*, les résultats des quatre mesures suivantes ont été comparés :

- $Coût$ : la mesure de coût proposée dans [59] pour calculer la similarité structurelle entre deux modèles de processus.
- $QLing$ : la mesure basée sur la notion de quantificateurs linguistiques présentée dans la sous-section 3.5.4.1 page 96.
- $CBip(i)$ : la mesure basée sur la première condition bipolaire proposée dans la sous-section 3.5.4.2 page 98.
- $CBip(ii)$ : la mesure basée sur la deuxième condition bipolaire présentée dans la sous-section 3.5.4.2 page 98.

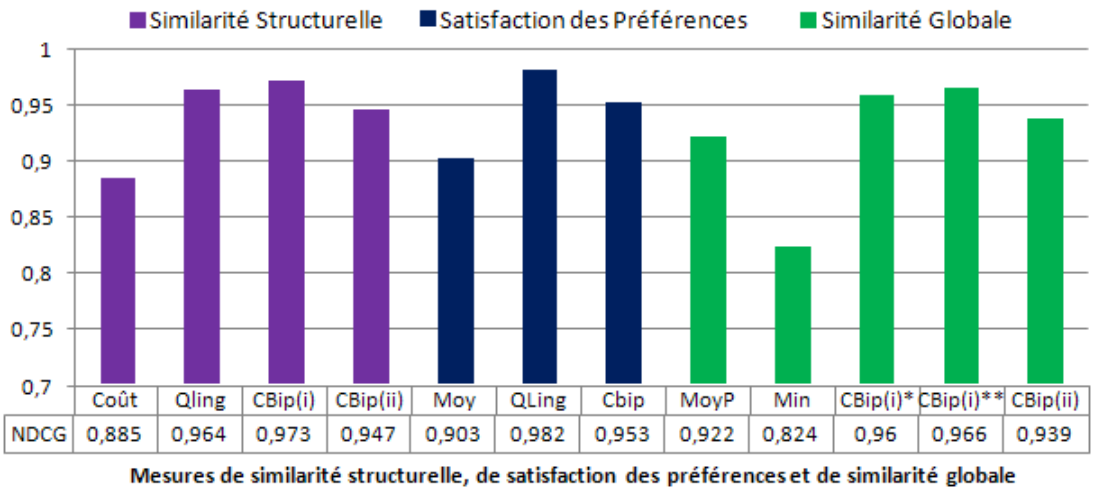
Concernant la catégorie, *Satisfaction des Préférences*, une comparaison entre trois mesures d'évaluation a été réalisée :

- $Moy$ : la mesure basée sur la moyenne des degrés de satisfaction proposée dans la sous-section 3.5.3.1 page 92.
- $QLing$ : la mesure basée sur les quantificateurs linguistiques tel que présenté dans la sous-section 3.5.3.2 page 92.
- $CBip$ : la mesure basée sur des préférences de qualité bipolaires présentée en sous-section 3.5.3.3 page 94.

Enfin, la catégorie, *Similarité Globale*, présente une comparaison entre les cinq mesures suivantes :

- $MoyP$ : la mesure sur la moyenne pondérée entre le degré de similarité structurelle et le degré de satisfaction globale des préférences de qualité. Voir la sous-section 3.5.5.1 page 100.
- $Min$ : la mesure basée sur l'opérateur « min » appliqué entre le degré de similarité structurelle et le degré de satisfaction des préférences.
- $CBip(i)^*$ : la troisième mesure d'agrégation de la sous-section 3.5.5 page 100 combinée avec la première condition bipolaire concernant la similarité structurelle présentée dans la sous-section 3.5.4.2 page 98.
- $CBip(i)^{**}$ : la troisième mesure d'agrégation de la sous-section 3.5.5 page 100 combinée avec la deuxième condition bipolaire de la similarité structurelle proposée dans la sous-section 3.5.4.2 page 98.
- $CBip(ii)$ : la quatrième mesure d'évaluation de la similarité globale entre deux modèles de processus présentée dans la sous-section 3.5.5 page 100.

En analysant les résultats des trois catégories décrites ci-dessus, nous constatons que les mesures proposées dans cette thèse à l'exception de la mesure « $Min$ » ont fourni des degrés NDCG pertinents montrant leur efficacité et la pertinence de leurs résultats. Ces degrés sont meilleurs que ceux obtenus par les mesures n'ayant pas de sémantique explicite (i.e., Coût, Moy et MoyP). Ainsi, les mesures proposées dans cette thèse pour la sélection de modèles de processus correspondent le mieux aux attentes de l'utilisateur. Cependant, dû à son critère très restrictif, la mesure « $Min$ » a retourné des résultats non satisfaisants du point de vue de l'utilisateur malgré sa puissance sémantique associée




| | Coût | Qling | CBip(i) | CBip(ii) | Moy | QLing | Cbip | MoyP | Min | CBip(i)* | CBip(i)** | CBip(ii) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NDCG | 0,885 | 0,964 | 0,973 | 0,947 | 0,903 | 0,982 | 0,953 | 0,922 | 0,824 | 0,96 | 0,966 | 0,939 |

FIGURE 4.4 – Les degrés NDCG des mesures d'évaluation de modèles de processus.

aux résultats.

## 4.6 Conclusion

Nous avons présenté dans ce chapitre le prototype de sélection de modèles de processus en prenant en considération les exigences structurelles et de qualité de l'utilisateur. Ce prototype a servi comme support d'évaluation pour montrer l'efficacité et la pertinence d'une approche d'évaluation de modèles de processus prenant notamment en considération des préférences délivrant des résultats qualifiés. Il permet à l'utilisateur de spécifier sa requête et l'ensemble des modèles de processus cibles pour retourner par la suite les modèles les plus intéressants du point de vue de l'utilisateur. Pour cela, il exécute la procédure d'appariement et calcule le degré de similarité structurelle, de satisfaction des préférences de qualité et de la similarité globale en combinant les deux premiers degrés.

Le prototype étendu dans le cadre du projet AOC permet donc une évaluation flexible et subjective des requêtes de modèles de processus. Chaque modèle de processus est associé à trois degrés exprimant respectivement la satisfaction de la requête de l'utilisateur suivant : (i) l'aspect structurel, (ii) l'aspect de qualité de service et (iii) les deux aspects (i) et (ii) à la fois. Ainsi, les modèles de processus retournés sont ordonnés du plus intéressant au moins intéressant par rapport aux contraintes structurelles et de qualité de l'utilisateur. De plus, l'approche fondée sur les conditions floues a été prolongée de manière à exploiter les requêtes utilisateur exprimées à l'aide de quantificateurs linguistiques et de conditions bipolaires. Ceci a permis d'éviter les réponses vides dans le cas de requêtes très sélectives et de réduire l'espace de recherche en éliminant les modèles de processus qui ne vérifient pas les contraintes de l'utilisateur.

Nous avons également présenté l'analyse de la complexité de l'approche proposée et montré que la complexité globale de l'évaluation des préférences est *polynomiale*, qui est négligeable comparée à la complexité exponentielle de l'algorithme d'appariement structurel, en général.

Enfin, des expérimentations ont été réalisées et décrites pour évaluer l'efficacité et la pertinence de l'approche de sélection de modèles de processus proposée. À travers ces expérimentations, nous avons montré que le temps pris pour évaluer les préférences de qualité est insignifiant comparé au temps de l'appariement structurel. Ce résultat est très important, car ceci permet de réduire le nombre de modèles de processus à évaluer lorsque ces derniers ne satisfont pas les contraintes de qualité de l'utilisateur et ainsi réduire le nombre de modèles de processus à traiter par les algorithmes d'appariement structurel qui explosent très rapidement au passage à large échelle [1]. Ces expérimentations ont montré que les mesures d'évaluation proposées sont efficaces et pertinentes en retournant les modèles de processus les plus intéressants vis-à-vis de l'utilisateur avec une sémantique simple et claire.

1. L'expression « passage à large échelle » désigne la capacité d'un système à traiter de grands volumes d'informations.

# Chapitre 5

# Skyline par similarité dans le contexte des bases de données de graphes

## 5.1 Introduction

La recherche de graphes similaires à une requête à graphe constitue l'un des problèmes fondamentaux dans les Bases de Données de Graphes (BDG). Plusieurs approches ont été proposées pour traiter ce type de problèmes et répondre aux requêtes de recherche de graphes par similarité (voir [71, 119, 133, 146]). Toutes ces méthodes s'appuient, généralement, sur une seule mesure pour évaluer la similarité entre graphes, en prenant en considération les étiquettes de leurs nœuds et/ou de leurs arêtes. Toutefois, un graphe est une structure complexe et comprend une multitude de caractéristiques de base. Il est alors difficile d'avoir une interprétation claire de la similarité entre graphes en utilisant un seul scalaire, calculé par exemple à l'aide de l'une des mesures présentées en section 1.3.3 du chapitre 1. En effet, chacune de ces mesures a ses propres propriétés et ses propres avantages et inconvénients (voir la section 1.3.3).

Pour répondre à cette difficulté, nous pensons qu'il peut être nécessaire de manipuler simultanément plusieurs mesures ou indices captant différentes caractéristiques des graphes. Chaque indice évalue une distance (ou une similarité) locale liée à un aspect donné dans la structure du graphe. Ainsi, la similarité entre graphes est un vecteur de mesures de distances locales au lieu d'une seule mesure globale. De cette façon, on représente l'intégralité des informations fournies par les similarités et on affranchit l'utilisateur du choix d'une mesure complexe de similarité parmi celles proposées dans la littérature. Le problème est alors d'ordonner les graphes cibles à partir de leur vecteur de similarité.

Pour ce faire, nous proposons dans ce chapitre une nouvelle approche de traitement de requêtes par similarité que nous appelons « *skyline*[1] *par similarité* » [1, 4]. D'une manière générale, le skyline par similarité d'une requête à graphe est défini par le sous-ensemble des graphes de la base de données non-dominés au sens de Pareto. L'idée est d'effectuer une comparaison multidimension-

1. La notion de skyline est déjà introduite en sous-section 2.3.1 du chapitre 2.

nelle entre graphes en termes de « $d$ » mesures de distances locales et d'identifier les graphes qui sont maximalement similaires au sens d'*une relation de dominance par similarité*. Les principales contributions de ce travail sont :

1. introduction de la notion de similarité composée entre graphes ($SCG$) ;
2. définition de la relation de dominance par similarité entre graphes ;
3. proposition d'une définition formelle du skyline de graphes par similarité, i.e., les graphes de la base non-dominés au sens de Pareto, en se basant sur la relation de dominance par similarité définie auparavant ;
4. définition d'une méthode de raffinement permettant de réduire la taille du skyline, qui est souvent très importante, en extrayant un sous-ensemble de graphes non-dominés qui est aussi divers que possible, mais dont la taille est raisonnable.

Dans ce qui suit, la section 5.2 introduit la notion de skyline par similarité dédiée aux requêtes à graphes. Dans les sections 5.3 et 5.4, nous décrivons respectivement la méthodologie d'évaluation des requêtes à graphes et le prototype préliminaire développé pour appliquer notre approche. Ce système calcule lors de l'évaluation d'une requête utilisateur, l'ensemble des graphes de la base de données *les plus similaires* au graphe de la requête retournés par le skyline. Nous proposons également, en section 5.5, une méthode de raffinement pour réduire le nombre de réponses retournés dans le skyline en utilisant la notion de *diversité*. Enfin, la section 5.6 conclut le chapitre.

## 5.2 Skyline de graphes par similarité

Dans cette section, nous définissons le concept, appelé *skyline par similarité*, pour traiter le problème de recherche de graphes par similarité sans avoir besoin de spécifier une mesure de similarité globale entre les graphes. Pour cela, nous considérons que la similarité entre chaque graphe de la base de données interrogée et le graphe requête, est une notion composée, i.e., un vecteur de mesures de distances. En d'autres termes, la similarité entre deux graphes est un vecteur de $d$ indices au lieu d'un seul scalaire, où chaque indice est calculé à l'aide d'une mesure de distance locale exprimant la mesure dans laquelle les deux graphes sont similaires par rapport à certains aspects et caractéristiques. La définition formelle de la similarité composée entre graphes peut être écrite :

**Définition 5.2.1 (Similarité composée entre graphes, SCG)**. Soient $g$ et $g'$ deux graphes. Une **similarité composée** entre $g$ et $g'$ est un vecteur de mesures de distance, notée

$$SCG\left(g, g'\right) = \left(Dist_1\left(g, g'\right), Dist_2\left(g, g'\right), \ldots, Dist_d\left(g, g'\right)\right) \tag{5.1}$$

où $Dist_i\left(g, g'\right)$, pour $i = 1, ..., d$, dénote une mesure de distances entre $g$ et $g'$.

Considérons, dans ce qui suit, $D = \{g_1, g_2, ..., g_n\}$ une base de données composée de $n$ graphes cibles et $q$ une requête de recherche de graphes par similarité (ou simplement, une requête par similarité).

L'idée est de procéder à une comparaison multidimensionnelle en terme de $d$ mesures de distances, pour rechercher les graphes de la base $D$ qui sont maximalement similaires au graphe requête $q$ au sens de la relation de dominance par similarité définie ci-dessous :

**Définition 5.2.2 (Relation de dominance par similarité)**. Soient une requête à graphe $q$ et deux graphes cibles $g$ et $g'$. On dit que le graphe $g$ **domine par similarité** le graphe $g'$ dans le contexte de la requête $q$, noté $g \succ_q g'$, si et seulement si les deux conditions suivantes sont vérifiées :

1. $\forall i \in \{1, ..., d\}$, $Dist_i(g, q) \leq Dist_i(g', q)$,
2. $\exists k \in \{1, ..., d\}$, $Dist_k(g, q) < Dist_k(g', q)$.

En d'autres termes, la relation $g \succ_q g'$ est vérifiée si (i) $g$ est au moins aussi similaire à la requête $q$ que $g'$ sur toutes les dimensions et (ii) $g$ est (strictement) plus similaire à $q$ que le graphe $g'$ sur au moins une dimension. Ainsi, le graphe $g$ est potentiellement plus intéressant que $g'$ comme graphe réponse à la requête de l'utilisateur, en terme de similarité. En conséquence, les graphes les plus similaires à $q$ sont ceux qui ne sont dominés (au sens de la définition (5.2.2)) par aucun autre graphe de la base de données $D$. L'ensemble de ces graphes, appelés *graphes optimaux au sens de Pareto*, représente ce que nous appelons *le skyline de graphes par similarité*, noté $SGS$ :

$$SGS(D, q) = \left\{ g \in D | \nexists g' \in D, g' \succ_q g \right\} \tag{5.2}$$

où $g' \succ_q g$ signifie que le graphe $g$ est dominé par similarité par le graphe $g'$ dans le contexte de la requête $q$.

Il est important de noter que les $d$-mesures de distance doivent être dans la mesure du possible non-redondantes au regard de leurs valeurs de similarité entre graphes. La distance $Dist_i$ est redondante avec la distance $Dist_j$ si et seulement l'implication suivante est tout le temps vérifiée, pour $i, j = 1, ..., d$ et $i \neq j$ :

$$Dist_i(g, q) \leq Dist_i(g', q) \Rightarrow Dist_j(g, q) \leq Dist_j(g', q)$$

avec $SCG(Dist_1(g_i, q), ..., Dist_n(g_i, q))$ et $SCG(Dist_1(g_j, q), ..., Dist_n(g_j, q))$ sont les vecteurs de mesures de distances respectifs de $g$ et $g'$.

Pour cela, nous avons analysé les mesures de similarité présentées en sous-section 1.3.3 et constaté que la distance d'édition, la distance basée sur le sous-graphe commun maximal et celle basée sur l'union de graphes ne sont pas redondantes lorsqu'elles sont utilisées à la fois dans la définition du vecteur de similarité $SCG$ associé à une requête de graphe donnée. Quant à la distance basée sur le super-graphe commun minimal ($Dist_{SCM}$), elle est redondante avec la distance basée sur l'union de graphes ($Dist_{ug}$), c'est-à-dire, pour deux graphes $g$ et $g'$ comparés avec une requête $q$, si $Dist_{ug}(g, q) \geq Dist_{ug}(g', q)$ alors forcément $Dist_{SCM}(g, q) \geq Dist_{SCM}(g', q)$. En conséquence, la présence conjointe des deux mesures, $Dist_{ug}$ et $Dist_{SCM}$, dans le vecteur des distances n'est

pas nécessaire car l'information extraite de l'une d'entre elles suffira dans la similarité composée de graphes.

Concernant, la distance basée sur le sous-graphe commun maximal ($Dist_{scm}$) et celle basée sur l'union de graphes ($Dist_{ug}$), nous avons constaté que lorsque l'on compare les distances de deux graphes $g$ et $g'$ par rapport à une requête $q$, l'implication suivante n'est pas toujours vraie : $Dist_{scm}(g,q) \geq Dist_{scm}(g',q) \Rightarrow Dist_{ug}(g,q) \geq Dist_{ug}(g',q)$. En effet, cette implication n'est pas vérifiée lorsque les conditions suivantes sont satisfaites :

- $\exists m, n \in \mathbb{N}$, telles que :
  - $|g| = |g'| + n$
  - $|scm(g,q)| = |scm(g',q)| + m$ et
  - $n > m$

où $|g|$ dénote la taille du graphe $g$ en terme du nombre d'arêtes et $scm(g,q)$ est le sous-graphe commun maximal entre $g$ et $q$.

## 5.3 Méthodologie d'évaluation de requêtes à graphes

Nous présentons dans cette section une méthodologie d'évaluation pour répondre aux requêtes à graphes par similarité [1, 4]. Cette méthodologie se base sur la relation de dominance par similarité présentée dans la section précédente.

Dans un premier temps, nous définissons l'architecture de la méthodologie et ses principales procédures pour évaluer une requête à graphe. Ensuite, nous présentons en détail les spécificités de chacune des phases et concluons la section par un exemple illustratif pour montrer l'utilité de l'approche proposée dans l'évaluation d'une requête à graphe.

### 5.3.1 Architecture de la méthodologie

L'idée que nous défendons dans cette section est que la méthodologie d'évaluation de requêtes de recherche de graphes par similarité que nous proposons, au regard des méthodologies existantes, permet la prise en considération des caractéristiques de plusieurs mesures de calcul de distances entres graphes. Cette méthodologie est composée principalement de deux procédures consistant respectivement en : (i) le calcul de la similarité composée entre graphes ($SCG$), i.e., le vecteur de distance entre graphes et (ii) l'évaluation du skyline par similarité ($SGS$).

Comme présenté dans la figure 5.1, la première procédure de notre méthodologie évalue le vecteur de distance ($SCG$) de chaque graphe de la base de données comparé au graphe de la requête. Cette procédure calcule les vecteurs de mesures de distances des graphes cibles (par exemple, la distance d'édition $Dist_{Ed}$, la distance basée sur le sous-graphe commun maximal $Dist_{scm}$, etc.) par rapport au graphe requête de l'utilisateur. Quant à la deuxième procédure, elle applique une relation de dominance par similarité sur l'ensemble des vecteurs de mesures de distances calculés dans la phase précédente, pour déterminer ainsi le skyline de graphes par similarité. Ce dernier constitue l'ensemble des graphes maximalement similaires au graphe requête de l'utilisateur, en considérant plusieurs

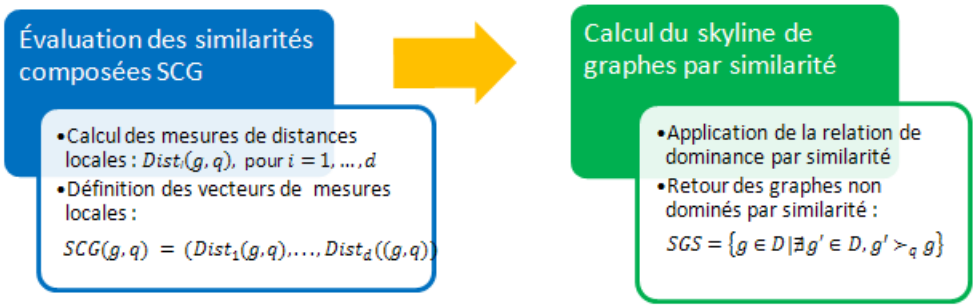



FIGURE 5.1 – Principales phases dans l'évaluation des requêtes à graphes.

mesures de calcul de distances plutôt qu'une seule mesure.

### 5.3.2 Processus d'évaluation des requêtes à graphes

Pour évaluer une requête de recherche de graphes par similarité, nous appliquons une méthodologie d'évaluation permettant de prendre en considération plusieurs mesures de similarité (ou de distances) entre graphes. Les avantages de celle-ci sont nombreux, parmi lesquels nous citons : (i) la possibilité d'affranchir l'utilisateur de choisir au hasard une mesure de calcul de distance entre graphes parmi celles proposées dans la littérature, (ii) la préservation des propriétés et des caractéristiques de chacune des mesures utilisées et enfin (iii) calcul des graphes les plus similaires au graphe de la requête de l'utilisateur qui est l'objectif principal de cette évaluation. Cette méthodologie est constituée des phases suivantes :

Considérons dans ce qui suit une requête de recherche de graphes par similarité $q$ adressée à une base de données de graphes $D = \{g_1, ..., g_n\}$.

**Phase 1 : Évaluation de la similarité composée** $SCG$

La première phase de notre méthodologie d'évaluation de requêtes à graphes consiste à calculer les similarités composées des graphes de la base de données interrogée. Pour chaque dimension $k = 1, ..., d$, la mesure de distance, $Dist_k(g_i, q)$, entre chaque graphe $g_i$ de la base de données $D$, pour $i = 1, ..., n$, et le graphe requête $q$, est calculée. Ceci permet donc de construire le vecteur des mesures de distance, $SCG(Dist_1(g_i, q), ..., Dist_d(g_i, q))$, de chaque graphe cible $g_i$, pour $i = 1, ..., n$.

Dans le cas où des mesures de distances basées sur le sous-graphe commun maximal ($scm$) et sur le super-graphe commun minimal ($MCS$) sont utilisées dans la composition du vecteur des mesures $SCG$, une procédure calculant ces deux critères (i.e., $scm$ et $SCM$) est d'abord exécutée avant l'évaluation de la similarité composée, comme l'illustre l'exemple de la sous-section 5.3.3. En effet, l'évaluation de certaines mesures de distance ne peut être effectuée sans l'information sur la taille du sous-graphe commun maximal, comme pour la mesure basée sur l'union de graphes (voir la sous-section 1.3.3.3), ou sur la taille du super-graphe commun minimal comme dans le cas de la mesure présentée en sous-section 1.3.3.4.

**Phase 2 : Calcul du skyline par similarité $SGS$**

La deuxième phase calcule le skyline de graphes par similarité $SGS$ correspondant au graphe requête de l'utilisateur. Pour ce faire, la relation de dominance par similarité présentée dans la définition 5.2.2 est appliquée sur l'ensemble des vecteurs de mesures de distance. Ainsi, le skyline composé des graphes maximalement similaires au graphe de la requête, c'est-à-dire, des graphes qui ne sont dominés par aucun autre graphe de la base de données suivant leur vecteur de mesures de distances, est retourné comme réponse à la requête de l'utilisateur.

Dans l'algorithme 5.1, nous présentons les étapes principales pour évaluer le skyline de graphes par similarité.

Il est intéressant de préciser que notre approche de recherche de graphes par similarité peut être intégrée dans un système d'interrogation de base de données de graphes de différents domaines d'application. Toutefois, il est important d'identifier à quel niveau, dans les étapes requises pour le traitement d'une requête à graphe (voir la section 1.4 du chapitre 1), l'évaluation du skyline par similarité est la plus pertinente.

L'évaluation des similarité composées $SCG$ entre les graphes de la base de données interrogée et la requête de l'utilisateur, nécessite de mesurer plusieurs distances. De plus, le calcul du skyline en général dépend à la fois du nombre des dimensions et de celui des objets à évaluer. Dans notre cas, les dimensions sont les distances à calculer et les objets représentent les vecteurs ou plus précisément les graphes à évaluer. Ainsi, pour améliorer la performance du système d'interrogation de bases de données de graphes, l'évaluation du skyline par similarité peut être appliquée après la phase de filtrage, i.e., à la phase de vérification, permettant ainsi de réduire le nombre de graphes candidats à son calcul.

### 5.3.3 Un exemple détaillé

Pour illustrer notre méthodologie de recherche de graphes par similarité, nous présentons dans ce qui suit un exemple détaillé où le nombre de dimensions $d$ est égal à 3. La similarité composée entre un graphe cible $g$ et une requête $q$, notée $SCG(g, q)$, est dans ce cas un vecteur de trois composantes exprimées en termes de mesures de distances présentées en sous-section 1.3.3 du chapitre 1, à savoir, la distance d'édition ($Dist_{Ed}$), la distance basée sur le sous-graphe commun maximal ($Dist_{scm}$) et la distance basée sur l'union de graphes ($Dist_{ug}$), i.e.,

$$SCG(g, q) = (Dist_{Ed}(g, q), Dist_{scm}(g, q), Dist_{ug}(g, q)).$$

Soient $D = \{g_1, g_2, g_3, g_4, g_5, g_6, g_7\}$ une base de données de graphes et $q$ une requête par similarité telles que présentées dans la figure 5.2. En supposons que la taille [2] d'un graphe est le nombre des arêtes le composant, nous déduisons à partir de la figure 5.2 que : $|g_1| = 6$, $|g_2| = 7$, $|g_3| = 7$, $|g_4| = 6$, $|g_5| = 8$, $|g_6| = 9$, $|g_7| = 10$ et $|q| = 6$.

2. À titre de rappel, la taille d'un graphe peut être exprimée par soit le nombre de ses nœuds, soit le nombre de ses arêtes soit les deux à la fois. Voir la définition (1.2.1) du chapitre 1.

**Algorithme 5.1** SGS.

**Entrée :** L'ensemble des graphes cibles candidats $D = \{g_1, ..., g_n\}$ et un graphe requête $q$.
**Sortie :** Skyline $SGS$ des graphes les plus similaires au graphe requête $q$.
**Début**

// Initialisation du skyline par similarité $SGS$ à l'ensemble $D$
$SGS \leftarrow D$
// Calcul des similarités composées $SCG(Dist_1(g_i, q), ..., Dist_d(g_i, q))$, noté $SCG_i$,
// de chaque graphe cible $g_i$ comparé à $q$
**Pour chaque** graphe $g_i$ de $D$ **faire**
  comparer $g_i$ à $q$
  **Pour** $k$ allant de 1 à $d$ **faire**
    // Calcul des distances $Dist_k$
    calculer $Dist_k(g_i, q)$
    // Composer le vecteur $SCG_i$ des distances du graphe $g_i$
    $SCG_i[k] \leftarrow Dist_k(g_i, q)$
  **Fin pour ;**
**Fin pour ;**
// Évaluation du skyline par similarité $SGS$
Soit $D'$ une variable temporaire contenant l'ensemble des graphes candidats de $D$
$D' \leftarrow D$
**Pour chaque** paire de graphe $(g_i, g_j) \in D^2$ et $i \neq j$ **faire**
  //Application de la relation de dominance par similarité
  comparer $SCG_i$ à $SCG_j$
  **Si** $g_i$ domine $g_j$ **alors**
    $SGS \leftarrow SGS \setminus \{g_j\}$
    $D' \leftarrow D' \setminus \{g_j\}$
    **Sinon Si** $g_j$ domine $g_i$ **alors**
      $SGS \leftarrow SGS \setminus \{g_i\}$
      $D' \leftarrow D' \setminus \{g_j\}$
    **Fin si ;**
  **Fin si ;**
**Fin pour ;**

Retourner $SGS$

**Fin**

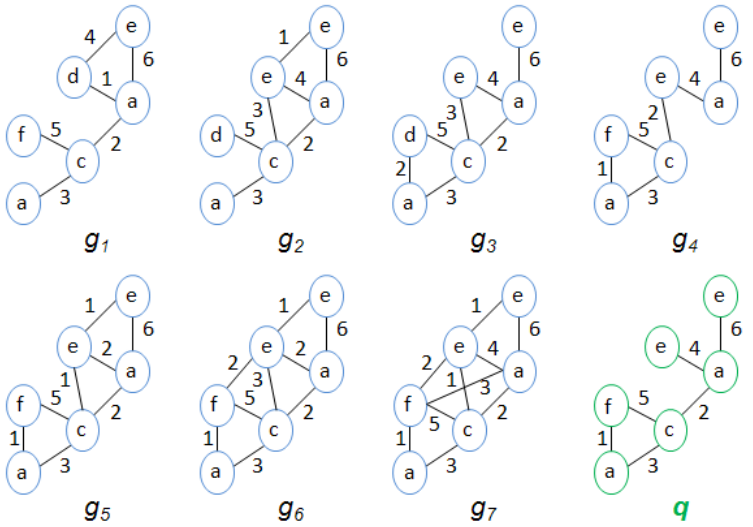



FIGURE 5.2 – Une base de données de graphes $D$ et une requête à graphe $q$.

TABLE 5.1 – Informations sur $|scm(g_i, q)|$, pour $i = 1, ..., 7$.

| Paire de graphes $(g_i, q)$ | $\lvert scm(g_i, q)\rvert$ |
|---|---|
| $(g_1, q)$ | 4 |
| $(g_2, q)$ | 4 |
| $(g_3, q)$ | 4 |
| $(g_4, q)$ | 3 |
| $(g_5, q)$ | 5 |
| $(g_6, q)$ | 5 |
| $(g_7, q)$ | 6 |

Afin de retourner les réponses les plus intéressantes à l'utilisateur suivant sa requête $q$, nous calculons le skyline de graphes par similarité $SGS(D, q)$ correspondant à la requête $q$ et à la base de données de graphes $D$. Pour ce faire, nous procédons comme suit :

1. **Calcul de la taille du sous-graphe commun maximal** $|scm(g_i, q)|$

   En première étape, nous évaluons la taille (i.e., le nombre d'arêtes) du sous-graphe commun maximal, $|scm(g_i, q)|$, entre chaque graphe $g_i$ de la base $D$ et le graphe requête $q$, pour $i = 1, ..., 7$. Les valeurs de $|scm(g_i, q)|$, pour $i = 1, ..., 7$, sont présentées dans le tableau 5.1.

2. **Évaluation de la similarité composée** $SCG(g_i, q)$

   Ensuite, nous déterminons les vecteurs de similarité entre chaque graphe $g_i$ de la base $D$ et la requête $q$, $SCG(g_i, q)$, pour $i = 1, ..., 7$, en calculant leur distance d'édition $Dist_{Ed}(g_i, q)$, leur distance basée sur le sous-graphe commun maximal, $Dist_{scm}(g_i, q)$ et leur distance basée sur l'union de graphes, $Dist_{ug}(g_i, q)$. Le tableau 5.2 résume l'ensemble de ces vecteurs de similarité.

TABLE 5.2 – Vecteurs de similarité entre les graphes de la base $D$ et la requête $q$.

| Paire de graphes $(g_i, q)$ | $Dist_{Ed}(g_i, q)$ | $Dist_{scm}(g_i, q)$ | $Dist_{ug}(g_i, q)$ |
|---|---|---|---|
| $(g_1, q)$ | **4** | ***0,33*** | **0,50** |
| $(g_2, q)$ | 4 | 0,43 | 0,56 |
| $(g_3, q)$ | 3 | 0,43 | 0,56 |
| $(g_4, q)$ | ***2*** | **0,50** | **0,67** |
| $(g_5, q)$ | ***3*** | ***0,38*** | ***0,44*** |
| $(g_6, q)$ | 4 | 0,44 | 0,50 |
| $(g_7, q)$ | **4** | **0,40** | ***0,40*** |

3. **Détermination du skyline par similarité** $SGS(D, q)$

   Enfin, par application de la définition (5.2.2), l'ensemble des graphes optimaux au sens de Pareto, i.e., le skyline de graphes par similarité, est donné par :

   $SGS(D, q) = \{g_1, g_4, g_5, g_7\}$.

   À partir des vecteurs de similarité présentés dans le tableau 5.2, il est facile de vérifier que (i) le graphe cible $g_2 \notin SGS(D, q)$ car il est dominé par $g_1$, $g_3$, $g_5$ et $g_7$, (ii) le graphe $g_3 \notin SGS(D, q)$ parce que $g_5$ le domine et enfin (iii) le graphe $g_6 \notin SGS(D, q)$ car il est dominé par $g_1$, $g_5$ et $g_7$.

Ainsi, les graphes de la base de données $D$ maximalement similaires au graphe requête $q$ sont $g_1$, $g_4$, $g_5$ et $g_7$. En effet, en analysant les graphes réponses retournés et leurs vecteurs de similarité, nous constatons, d'une part, qu'il n'y a pas de redondance d'informations entre les mesures de distance et d'autre part que :

- le graphe $g_1$ est le plus intéressant par rapport à la mesure de distance $Dist_{scm}$. Cela est dû aux raisons suivantes : (i) $g_1$ satisfait un maximum de caractéristiques requises par la requête $q$ que d'autres graphes de même taille et (ii) $g_1$ et $q$ sont de même taille. Toutefois, $g_1$ est moins intéressant au regard des caractéristiques manquantes et superflues (i.e., $Dist_{Ed}$) ;
- le graphe $g_4$ représente le meilleur graphe cible de la base de données $D$ suivant la mesure de distance $Dist_{Ed}$. Cela signifie qu'il est le plus intéressant suivant le nombre de désaccords avec le graphe requête $q$. D'autre part, $g_4$ est moins satisfaisant par rapport aux concordances avec $q$ au sens de la notion « sous-graphe commun maximal, *scm* » ;
- le graphe $g_7$ est le plus intéressant suivant la mesure de distance $Dist_{ug}$. Cela est dû au fait que le graphe requête $q$ est inclus dans $g_7$, i.e., $g_7 \supset q$. Néanmoins, il est moins intéressant suivant les caractéristiques manquantes et superflues (i.e., $Dist_{Ed}$) ;
- enfin, le graphe $g_5$ peut être considéré comme étant un bon compromis entre les trois mesures de distances $Dist_{Ed}$, $Dist_{scm}$ et $Dist_{ug}$.

Examinons maintenant les résultats obtenus en utilisant seulement une seule mesure de calcul de distance entre chaque graphe de la base $D$ et le graphe requête $q$. Si l'utilisateur s'intéresse au $k$ (= 3) meilleures graphes suivant la mesure basée sur la distance d'édition (i.e., $Dist_{Ed}$), $g_3$ est retourné

comme graphe réponse à la requête $q$. Tandis qu'en utilisant l'approche basée sur le skyline par similarité, le graphe $g_3$ est éliminé de l'ensemble des graphes réponses car $g_5$ est meilleur suivant les deux autres mesures de distance, c'est-à-dire, la distance basée sur le sous-graphe commun maximal (i.e., $Dist_{scm}$) et celle basée sur l'union de graphes (i.e., $Dist_{ug}$).

À partir de cet exemple, il est facile de vérifier que la complexité temporelle de notre approche de skyline par similarité dépend principalement du temps d'exécution des deux fonctions : (i) la fonction calculant les distances entre les graphes de la base de données interrogée et le graphe requête et, (ii) la fonction appliquant la relation de dominance sur les vecteurs de similarité $SCG(g_i, q)$, pour $i = 1, ..., n$.

Les mesures basées sur le sous-graphe commun maximal pour calculer la distance entre deux graphes, telles que la distance basée sur le sous-graphe commun maximal $Dist_{scm}$ et celle basée sur l'union de graphes $Dist_{ug}$, leurs temps d'exécution dépendent principalement du temps de recherche du sous-graphe commun maximal entre chaque graphe de la base de données et le graphe requête, qui également un problème NP-complet [35, 34]. Néanmoins, des méthodes ont été proposées dans différents domaines d'application permettant de réduire cette complexité en un temps polynomial [37, 38]. Il existe également des algorithmes dédiés aux grandes bases de données pour déterminer le sous-graphe commun maximal entre chaque deux graphes comparés [102, 34, 35].

Par ailleurs, la distance d'édition est un problème NP-complet [153]. En conséquence, comme tout autre système d'interrogation de bases de données évaluant la distance d'édition entre deux graphes, notre approche de skyline par similarité est beaucoup plus adaptée dans le cas des bases de données de graphes de petite taille.

## 5.4 Prototype de recherche de graphes par similarité

Pour valider notre approche d'interrogation de bases de données de graphes, nous présentons dans cette section le prototype préliminaire développé, appelé « *GraphSim*[3] », permettant de répondre aux requêtes à graphes par similarité. Ce prototype est une implémentation de l'approche « Skyline de graphes par similarité » décrite en section 5.2. Il reçoit en entrée les graphes cibles de la base de données interrogée et le graphe requête de l'utilisateur, pour retourner ensuite l'ensemble des graphes les plus similaires au graphe de la requête. Pour cela, *GraphSim* s'appuie sur deux mesures de distance, à savoir, la distance basée sur le sous-graphe commun maximal ($Dist_{scm}$) et la distance basée sur l'union de graphes ($Dist_{ug}$) et applique la relation de dominance par similarité entre graphes introduite dans la définition (5.2.2).

Comme indiqué dans les deux formules (1.3) et (1.4) du chapitre 1, l'évaluation de la distance basée sur le sous-graphe commun maximal et de celle basée sur l'union de graphes s'appuie essentiellement sur le calcul de la taille du sous-graphe commun maximal entre chaque paire de graphes comparés par le prototype « *GraphSim* ». Concernant la distance d'édition présentée dans la sous-section 1.3.3 du chapitre 1, cette mesure n'a pas été intégrée dans notre prototype, car non seulement

3. GraphSim est l'acronyme de « Graph Similarity » en anglais.

le problème du calcul de la distance d'édition entre graphes est NP-complet mais aussi l'utilisation des méthodes [52, 109, 115, 154] proposées dans la littérature pour évaluer une distance d'édition approximative entre les graphes cibles et le graphe requête pourrait fortement impacter les résultats du skyline par similarité.

Le temps d'exécution du processus d'évaluation de requête à graphe dans « GraphSim » dépend ainsi du temps d'exécution de la fonction « $F_1$ » évaluant le sous-graphe commun maximal entre deux graphes et de la fonction « $F_2$ » calculant le skyline par similarité en appliquant la relation de dominance au sens de Pareto. La complexité de cette dernière est de l'ordre $n^2$ (i.e., polynomiale). En conséquence, la complexité temporelle globale dépend étroitement de celle de la fonction « $F_1$ » qui peut être réduite en un temps polynomial dans certains domaines d'application [37, 38]. Cette fonction de calcul de sous-graphe commun maximal « $F_2$ » peut être également améliorée par un algorithme fonctionnant sur des grandes bases de données de graphes, comme proposé dans [102, 34, 12].

**i) Architecture du prototype**

Dans le cadre de l'interrogation de bases de données de graphes, nous proposons l'application « *GraphSim* », pour retrouver tous les graphes de la base de données *les plus similaires* à une requête à graphe au sens de la relation de dominance par similarité définie dans la section 5.2. Cette relation se fonde sur deux mesures de distance entre graphes non-redondantes : (i) la distance basée sur le sous-graphe commun maximal ($Dist_{scm}$) et (ii) la distance basée sur l'union de graphes ($Dist_{ug}$).

L'architecture générale de *GraphSim* est présentée dans la figure 5.3. Elle est composée des modules suivants :

- **Le module des scm** : Ce module permet d'effectuer un pré-calcul avant l'évaluation du skyline de graphes par similarité proposée dans notre approche. Ce pré-calcul consiste en premier lieu en l'identification du sous-graphe commun maximal, noté $scm(g_i, q)$, entre chaque graphe $g_i$ de la base de données interrogée et le graphe requête $q$. Ensuite, il détermine la taille de ce sous-graphe, c'est-à-dire, $|scm(g_i, q)|$, en calculant, dans notre cas, le nombre d'arêtes le composant.
- **Le module des distances :** Ce module a pour objectif (i) de calculer pour chaque graphe $g_i$ de la base de données, sa similarité composée $SCG(g_i, q)$ par rapport au graphe requête $q$ et (ii) de sauvegarder cette dernière dans **le répertoire des vecteurs de distances**. Pour cela, il évalue les mesures de distance entre $g_i$ et le graphe $q$, à savoir, la distance basée sur le sous-graphe commun maximal ($Dist_{scm}(g_i, q)$) et la distance basée sur l'union de graphes ($Dist_{ug}(g_i q)$).
- **Le module du Skyline :** Ce module applique la relation de dominance par similarité proposée, en comparant les vecteurs de distances locales (i.e., $SCG(g_i, q)$) des graphes de la base de données pour sélectionner ensuite ceux qui ne sont dominés par aucun autre graphe. Il permet de retourner les graphes maximalement similaires au graphe de la requête en se basant sur deux mesures de distances entre graphes, préservant ainsi les caractéristiques de chacune

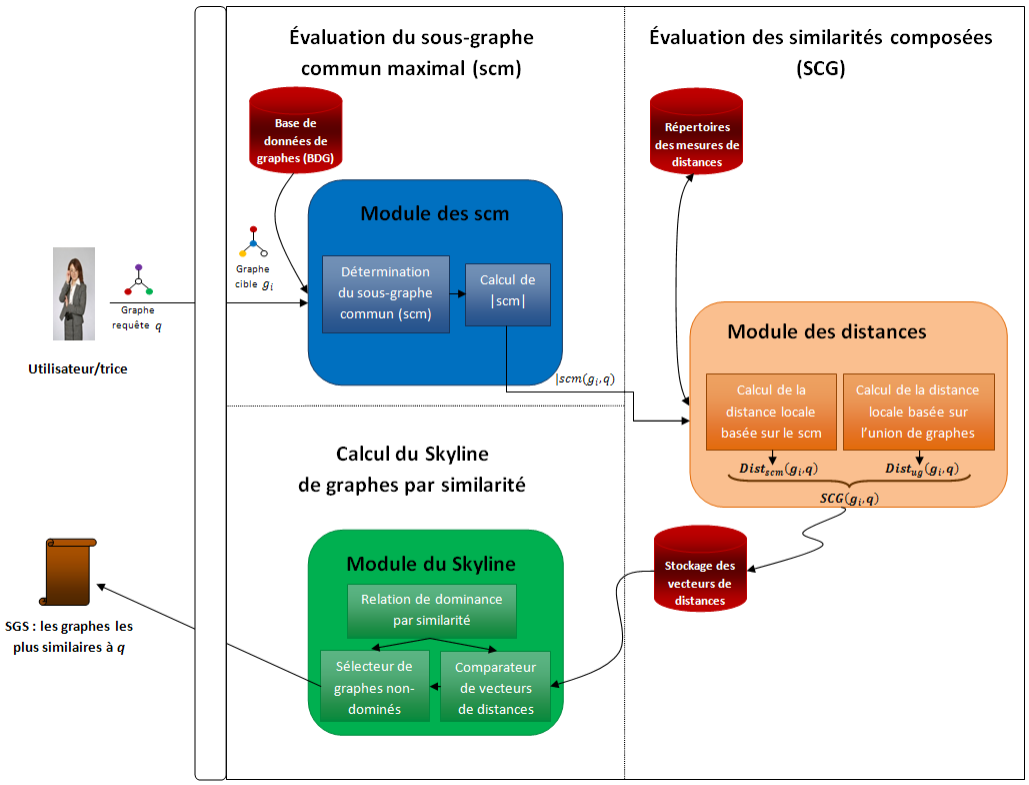



FIGURE 5.3 – Architecture générale du prototype *GraphSim*.

d'entre elles.

En plus des trois modules cités ci-dessus, *GraphSim* dispose d'une interface permettant à l'utilisateur de créer sa base de données de graphes et d'introduire son graphe requête.

### ii) Description du prototype

Le prototype « *GraphSim* » développé dans cette thèse, a pour objectif de montrer la faisabilité de notre approche dans l'évaluation des requêtes de recherche de graphes par similarité, en se basant sur les mesures de distance $(Dist_{scm}, Dist_{ug})$. Il dispose d'une interface graphique permettant à l'utilisateur de spécifier sa requête et l'ensemble des graphes cibles à évaluer, pour retourner par la suite les graphes les plus similaires à la requête ainsi que leur distance basée sur le sous-graphe commun maximal et leur distance basée sur l'union de graphes.

*GraphSim* est une application se basant sur le système PostgreSQL, permettant de créer des bases de données hétérogènes où les graphes cibles sont stockés. En d'autres termes, les bases de données créées par notre application peuvent contenir des graphes de différents types, à savoir, des graphes étiquetés non-orientés et des graphes non-étiquetés orientés. Pour éviter des comparaisons inutiles, l'application filtre les graphes de même type que le graphe requête pour évaluer leurs vecteurs de distances. En conséquence, *GraphSim* est une application qui peut être utilisée dans plusieurs domaines d'application modélisant les objets complexes par des graphes.

## 5.5 Raffinement de skyline de graphes par similarité

Un des problèmes qui peut survenir lors du calcul du skyline par similarité $SGS$ et d'un skyline en général, est sa taille qui est souvent très importante. Ainsi, il est difficile à l'utilisateur de choisir les meilleurs graphes parmi ceux retournés. Pour cela, il est souhaitable, du point de vue de l'utilisateur, de disposer d'un critère pertinent permettant de sélectionner un sous-ensemble, de taille raisonnable, des graphes les plus intéressants parmi ceux du skyline $SGS$.

Une solution à ce problème est d'utiliser le critère de diversité emprunté du domaine des systèmes de recommandation [103] et du domaine de recherche à base de cas [73]. Elle vise à sélectionner un sous-ensemble de graphes qui est *aussi divers que possible*. Ceci permet donc de fournir à l'utilisateur *une image globale* de l'ensemble des éléments de $SGS$.

Nous présentons, dans cette section, une méthode de raffinement du skyline de graphes par similarité pour retourner à l'utilisateur un sous-ensemble de graphes de taille (raisonnable) $k$ aussi divers que possible, parmi ceux qui sont maximalement similaires à son graphe requête.

### 5.5.1 Méthode de raffinement

Soit $S$ un sous-ensemble du skyline par similarité $SGS$. La propriété de diversité sur le sous-ensemble $S$ signifie que les graphes qu'il contient doivent être le plus dissimilaires possible. L'objectif est donc d'extraire à partir du skyline par similarité $SGS$, un sous-ensemble $\mathbb{S}$ de taille $k$ avec *une diversité maximale*, où $k$ est un paramètre défini par l'utilisateur pour répondre à ses besoins et à ses objectifs.

En s'inspirant des travaux de Kukkonen et Lampinen [87] proposés dans le domaine de l'optimisation multi-objectif, nous définissons la diversité d'un sous-ensemble de graphes $S$ ($\subseteq SGS$) comme suit :

**Définition 5.5.1 (Diversité d'un sous-ensemble de graphes $S$).** Soit $S$ un sous-ensemble du skyline par similarité $SGS$ composée de $k$ graphes maximalement similaires à la requête $q$. La diversité de $S$ est exprimée par un vecteur

$$Div(S) = (v_1, v_2, ..., v_d) \tag{5.3}$$

tel que :

$$v_i = \min\left\{Dist_i(g, g') | g, g' \in S\right\} \tag{5.4}$$

où $Dist_i(g, g')$ dénote une mesure de distance entre les graphes $g$ et $g'$. La valeur $v_i$ exprime la diversité sur la i$^{ième}$ dimension du sous-ensemble $S$.

Dans l'exemple présenté dans la sous-section 5.3.3, le vecteur de diversité sera donc composé de trois valeurs ($v_1$, $v_2$ et $v_3$) représentant respectivement la diversité suivant la distance normalisée de la distance d'édition (i.e., $Dist_1 = Dist_{N-Ed}$) en utilisant la fonction $f(x) = {}^{x}/_{(1+x)}$, la distance

basée sur le sous-graphe commun maximal (i.e., $Dist_2 = Dist_{scm}$) et la distance basée sur l'union de graphes (i.e., $Dist_3 = Dist_{ug}$). Par ailleurs, la normalisation de la distante d'édition n'est pas forcément nécessaire dans notre méthode pour calculer la diversité d'une sous-ensemble de « k » graphes.

Afin d'identifier le sous-ensemble $\mathbb{S}$ de diversité maximale à partir d'un skyline de graphes par similarité, noté $SGS$, les étapes suivantes sont appliquées :

**Étape 1 : Détermination des sous-ensembles $S_j \subseteq SGS$ de taille $k$**

Dans l'objectif de raffiner un skyline de graphes par similarité $SGS$, nous énumérons d'abord tous ses sous-ensembles candidats $S_j \subseteq SGS$ ($C^k_{|SGS|}$ sous-ensembles possibles et $j = 1, ..., C^k_{|SGS|}$) de taille $k$, où $k$ est le nombre de graphes réponses maximalement similaires au graphe requête souhaité par l'utilisateur (i.e., $|S_j| = k$).

**Étape 2 : Calcul des vecteurs de diversité $Div(S_j) = (v_1, ..., v_d)$**

Nous calculons par la suite le vecteur de diversités $Div(S_j) = (v_1, ..., v_d)$ de chaque sous-ensemble de graphes $S_j$, pour $j = 1, ..., C^k_{|SGS|}$, en appliquant la formule (5.4).

**Étape 3 : Ordonnancement décroissant des sous-ensembles $S_j$**

Ensuite, pour chaque dimension $i$ ($i = 1, ..., d$), nous ordonnons d'une manière décroissante tous les sous-ensembles candidats $S_j$ suivant leur diversité $v_i$. Ainsi, un rang de valeur égale à 1 sur une dimension donnée signifie que le sous-ensemble de graphes correspondant possède la meilleure valeur de diversité.

**Étape 4 : Évaluation de la diversité de chaque sous-ensemble candidat $S_j$**

Enfin, pour évaluer la valeur de la diversité globale, $Div_g$, de chaque sous-ensemble candidat $S$ *sur toutes les dimensions*, nous appliquons la formule suivante :

$$Div_g(S) = \sum_{i=1}^{d} rang_i(S)$$

Le sous-ensemble de graphes minimisant la somme de ses positions relativement à tous les rangs, est considéré comme le sous-ensemble ayant la diversité maximale. Ainsi, le sous-ensemble de graphes maximalement similaires à la requête de l'utilisateur avec une diversité maximale, noté $\mathbb{S}$, est caractérisé par

$$Div_g(\mathbb{S}) = \min_{S} Div_g(S)$$

où $S \subseteq SGS$ et $|S| = k$.

En conséquence, l'ensemble $\mathbb{S}$ des graphes maximalement similaires au graphe de la requête est retourné à l'utilisateur avec une diversité maximale, offrant une vue globale des graphes composant le skyline par similarité $SGS$.

L'algorithme 5.2 résume les principales étapes du processus de raffinement de skyline en se basant

TABLE 5.3 – Sous-ensembles candidats de graphes avec leur diversité.

| Sous-ensembles candidats | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|
| $S_1 = \{g_1, g_4\}$ | 0,86 | 0,67 | 0,80 |
| $S_2 = \{g_1, g_5\}$ | 0,83 | 0,50 | 0,60 |
| $S_3 = \{g_1, g_7\}$ | 0,87 | 0,60 | 0,67 |
| $S_4 = \{g_4, g_5\}$ | 0,80 | 0,62 | 0,73 |
| $S_5 = \{g_4, g_7\}$ | 0,83 | 0,70 | 0,77 |
| $S_6 = \{g_5, g_7\}$ | 0,75 | 0,50 | 0,61 |

sur le critère de diversité, dans lequel la variable $S_i.Div[r]$ représente la diversité du sous-ensemble $S_i$ sur la $r^{ième}$ dimension (i.e., $v_r$ dans la formule (5.3), pour $r = 1, ..., d$) ; $S_i.Div_g$ est la diversité globale $Div(S_i)$ de $S_i$ et $S_i.rang[r]$ dénote le rang du sous-ensemble $S_i$ sur la $r^{ième}$ dimension.

### 5.5.2 Un exemple illustratif

Dans cette sous-section, nous présentons un exemple détaillé pour illustrer notre approche de raffinement de skyline de graphes par similarité.

Reprenons le skyline de graphes $SGS(D, q) = \{g_1, g_4, g_5, g_7\}$ obtenu dans l'exemple illustratif de la sous-section 5.3.3 (voir la figure 5.2 page 130). Supposons maintenant que l'utilisateur est intéressé par les k (= 2) meilleurs graphes du skyline par similarité $SGS$ suivant le critère de diversité. Pour cela, nous appliquons les étapes de l'approche de raffinement comme suit :

1. **Détermination des sous-ensembles $S_j \subseteq SGS$ de taille $k = 2$**
   Par application de la première étape du processus de raffinement sur le skyline $SGS = \{g_1, g_4, g_5, g_7\}$, nous obtenons 6 sous-ensembles candidats de taille k (= 2). Voir la première colonne du tableau 5.3.

2. **Calcul des vecteurs de diversité $Div(S_j) = (v_1, v_2, v_3)$**
   En se basant sur la distance d'édition normalisée ($Dist_{N-Ed}$), la distance basée sur le sous-graphe commun maximal ($Dist_{scm}$) et la distance basée sur l'union de graphe ($Dist_{ug}$), nous évaluons pour chaque sous-ensemble de graphes $S_j$, pour $j = 1, ..., 6$, sa diversité $Div(S_j) = (v_1, v_2, v_3)$, telle que

   – $v_1 = \min \{Dist_{N-Ed}(g, g') | g, g' \in S_j\}$ ;
   – $v_2 = \min \{Dist_{scm}(g, g') | g, g' \in S_j\}$ ;
   – $v_3 = \min \{Dist_{ug}(g, g') | g, g' \in S_j\}$.

   Les valeurs de diversité $v_i$ de chaque sous-ensemble de graphes sont présentées dans le tableau 5.3.

3. **Ordonnancement décroissant des sous-ensembles $S_j$**
   Pour chaque dimension $i$, nous classons dans un ordre décroissant les sous-ensembles de graphes $S_j$ suivant leur valeur de diversité $v_i$, pour $i = 1, ..., 3$ et $j = 1, ..., 6$, comme présenté dans le

**Algorithme 5.2** RSD.

**Entrée :** Un skyline de graphes par similarité $SGS = \{g_1, ..., g_m\}$ et $k$ le nombre de graphes réponses souhaités.
**Sortie :** Un sous-ensemble de graphes $\mathbb{S} \subseteq SGS$ de diversité maximale.
**Début**

```
// Initialisation du sous-ensemble 𝕊 à l'ensemble vide
𝕊 ← Ø
// Énumération des sous-ensembles S_j ⊆ SGS de taille k
Soit S_SGS l'ensemble des sous-ensembles S_j ⊆ SGS de taille k
 calculer S_SGS
Pour chaque sous-ensemble S_i de S_SGS faire
   Pour r allant de 1 à d faire
      v_r ← min {Dist_r(g, g') | g, g' ∈ S_i}
      S_i.Div[r] ← v_r
   Fin pour ;
Fin pour ;
Pour chaque sous-ensemble S_i de S_SGS faire
   S_i.Div_g ← 0
   Pour r allant de 1 à d faire
      ordonner S_i suivant sa valeur v_r
      calculer S_i.rang[r]
       S_i.Div_g ← S_i.Div_g + S_i.rang[r]
   Fin pour ;
Fin pour ;
// Évaluation du skyline par similarité SGS
𝕊.Div_g ← S_1.Div_g
Pour chaque paire de graphe S_i ∈ S_SGS \ {S_1} faire
   //Choisir le sous-ensemble S_i de diversité maximale
   Si 𝕊.Div_g > S_i.Div_g alors
      𝕊.Div_g ← S_i.Div_g
      𝕊 ← S_i
   Fin si ;
Fin pour ;

Retourner 𝕊
```

**Fin**

TABLE 5.4 – Évaluation des sous-ensembles candidats de graphes.

| | $rang_1$ | $rang_2$ | $rang_3$ | $Div_g(S_j) = \sum_{i=1}^{3} rang_i$ |
|---|---|---|---|---|
| $S_1 = \{g_1, g_4\}$ | 2 | 2 | 1 | **5** |
| $S_2 = \{g_1, g_5\}$ | 3 | 5 | 6 | **14** |
| $S_3 = \{g_1, g_7\}$ | 1 | 4 | 4 | **9** |
| $S_4 = \{g_4, g_5\}$ | 4 | 3 | 3 | **10** |
| $S_5 = \{g_4, g_7\}$ | 3 | 1 | 2 | **6** |
| $S_6 = \{g_5, g_7\}$ | 5 | 5 | 5 | **15** |

tableau 5.4, où $rang_i$ dénote le rang de chaque sous-ensemble de graphes sur chaque dimension $i$ suivant sa diversité $v_i$.

4. **Évaluation de la diversité de chaque sous-ensemble candidat $S_j$**

   Enfin, pour évaluer la diversité globale de chaque sous-ensemble de graphes $S_j$ $(j = 1, ..., 6)$ sur toutes les dimensions, nous additionnons ses rangs $rang_i$, pour $i = 1, ..., 3$, calculés dans l'étape précédente. Voir la dernière colonne du tableau 5.4.

   À partir de ces résultats, nous constatons que $Div_g(S_1)$ est la plus petite valeur. Ainsi, l'ensemble aussi divers que possible de graphes maximalement similaires au graphe requête $q$ est $\mathbb{S} = S_1 = \{g_1, g_4\}$.

Les k (=2) meilleurs graphes suivant le critère de similarité et de diversité maximale retournés à l'utilisateur sont donc $g_1$ et $g_4$.

À partir de cet exemple, nous remarquons que l'étape pouvant être pénalisante dans la méthode de raffinement basée sur la diversité est l'étape 2 où les distances entre chaque paire de graphes de chaque sous-ensemble sur chaque dimension doivent être calculées. Autrement dit, sur chaque dimension $a_r$, pour $ra = 1, ..., d$, $C_k^2$ distances doivent être évaluées dans chaque sous-ensemble $S_i$ parmi $C_{|SGS|}^k$ sous-ensembles. Par ailleurs, il est important de noter qu'en pratique, le paramètre $k$ est généralement d'une valeur raisonnable qui n'axcède pas 4 ou 5. Cependant, pour améliorer le temps d'exécution de notre approche de raffinement, une méthode permettant d'éviter les calculs inutiles des distances entre graphes de certains sous-ensembles, est nécessaire.

## 5.6 Conclusion

Nous nous sommes intéressés dans ce chapitre à la problématique de l'évaluation de requêtes de recherche de graphes par similarité, dans le domaine de bases de données de graphes. Après avoir étudié les différentes mesures de calcul de similarité entre graphes les plus citées dans la littérature, nous avons constaté que chacune d'entre elles possède ses propres caractéristiques, ses propres avantages et inconvénients. Ainsi, l'utilisateur peut ne pas être en mesure de choisir la méthode de calcul de similarité la plus pertinente pour l'évaluation de sa requête à graphe. De plus,

il n'existe pas de mesure universelle qui peut être utilisée pour tout type de graphes et dans divers domaines d'application tels que le domaine de l'imagerie, la médecine, la chimie, etc.

Pour cela, nous avons proposé une approche permettant de retrouver l'ensemble des graphes les plus similaires au graphe de la requête, appelé *skyline de graphes par similarité*. Cette approche est basée sur *la relation de dominance* au sens de Pareto appliquée sur plusieurs mesures de calcul de distances (ou de similarités) entre graphes. Nous avons également décrit le processus d'évaluation de requêtes à graphes qui a été par la suite décrit et illustré par un exemple détaillé.

L'avantage de cette approche réside dans sa capacité à préserver l'information concernant la similarité sur différentes caractéristiques (i.e., différentes mesures de calcul de similarité), lorsque l'on compare deux graphes. Elle permet également d'épargner l'utilisateur à choisir une méthode de calcul de similarité parmi celles existantes, sachant qu'il n'existe pas de règles précises lui permettant de déterminer quelle mesure utiliser, dans quel cas et dans quel domaine d'application.

Pour montrer la faisabilité de notre méthode, nous avons présenté le prototype *GraphSim* (Graph Similarity). Nous avons décrit les différents modules implémentés permettant de calculer la similarité composée et le skyline d'une requête adressée à une base de données de graphes.

Enfin, nous avons proposé, dans ce chapitre, une méthode de raffinement permettant de réduire le nombre de graphes retournés dans le skyline par similarité lorsque ce dernier est de taille importante. Cette méthode se base sur *le critère de diversité* pour retourner un sous-ensemble *aussi divers que possible* de taille $k$ du skyline de graphes par similarité. Elle permet donc de retourner les $k$ meilleurs graphes réponses à la requête suivant le critère de similarité et de diversité maximales.

Nous nous intéressons dans le chapitre suivant au raffinement de skyline en général et proposons trois approches permettant de réduire la taille du skyline en sélectionnant les réponses les plus intéressantes suivant la requête de l'utilisateur. Le concept clé de ces approches est basé sur l'utilisation de quantificateurs linguistiques.

# Chapitre 6

# Raffinement de Skyline basé sur les quantificateurs linguistiques

## 6.1 Introduction

Les requêtes skyline représentent un paradigme très populaire et puissant pour extraire les objets ou les points (ou encore les éléments) les plus intéressants vis-à-vis de l'utilisateur, à partir d'un ensemble de données multidimensionnelles. Ces requêtes s'appuient sur le principe de dominance au sens de Pareto pour identifier l'ensemble des points (appelés aussi les points skyline) qui ne sont dominés par aucun autre point du même ensemble de données. Rappelons qu'un point $p$ *domine* (ou *est préféré à*) un autre point $p'$ si et seulement si $p$ est meilleur ou égal à $p'$ sur toutes les dimensions et strictement meilleur que $p'$ sur au moins une dimension. Pour plus de détails, voir la sous-section 2.4.1 du chapitre 2, où un état de l'art sur les requêtes skylines est présenté.

Dans le chapitre précédent, nous avons proposé une méthode de raffinement de skyline dans le contexte de recherche de graphes par similarité. Dans ce chapitre, nous nous intéressons au problème de raffinement de skyline en général. En effet, l'interrogation d'un ensemble de données de $d$-dimensions à l'aide de l'opérateur skyline, peut avoir comme résultat un nombre important de réponses, qui est généralement peu informatif du point de vue de l'utilisateur.

Diverses approches ont été proposées pour raffiner le skyline, c'est-à-dire réduire sa taille en un ensemble d'éléments de taille raisonnable [24, 25, 50, 53, 68, 76, 99, 110]. Cependant, ces approches considèrent que certains éléments (ou points) du skyline sont les plus intéressants par rapport à la requête, en appliquant des critères qui peuvent ne pas convenir à l'utilisateur. De plus, comme illustré à l'aide des exemples dans la sous-section 2.4.1.2 du chapitre 2, les éléments retournés par l'application de l'une de ces approches peuvent ne pas être les meilleures réponses à la requête de l'utilisateur. En d'autres termes, les approches de raffinement de skyline proposées dans la littérature ne permettent pas toujours de sélectionner les réponses les plus pertinentes aux attentes de l'utilisateur.

Pour cela, nous présentons dans ce chapitre une famille d'approches de raffinement basées sur la notion de quantificateur linguistique, pour réduire la taille du skyline [2, 3]. Plus précisément, les

approches que nous proposons se basent sur une relation de dominance plus sélective comparée à la relation de dominance classique au sens de Pareto, permettant ainsi de filtrer les points skyline qui répondent le mieux à la requête de l'utilisateur. Pour ce faire, les défis suivants sont à relever :

- introduction de critères permettant de discriminer les points skyline ;
- définition d'une relation de dominance plus sélective pour déterminer les meilleurs points du skyline par rapport à la requête de l'utilisateur ;
- ordonnancement des points skyline des plus intéressants au moins intéressants par rapport à la requête de l'utilisateur.

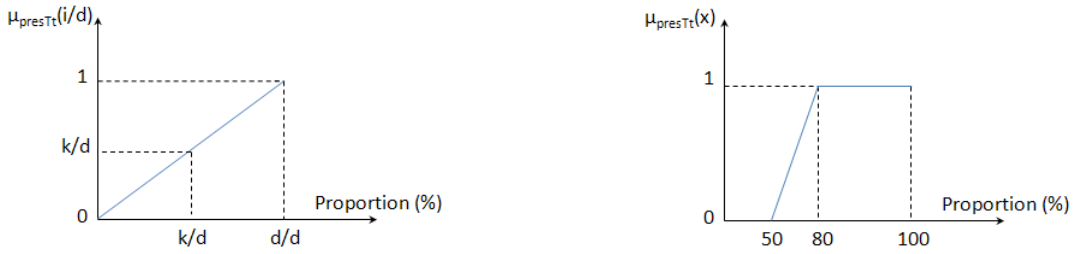
Dans ce qui suit, la section 6.2 introduit une relation de dominance graduée plus sélective que celle de Pareto, permettant une comparaison entre les éléments du skyline. Ensuite, trois méthodes de raffinement (RDD, MRM et RID) basées sur cette nouvelle relation de dominance sont présentées avec leurs algorithmes correspondants en section 6.3. Dans la section 6.4, nous décrivons un cadre illustrant l'application de ces approches de raffinement de skyline lorsque celui-ci est d'une taille assez importante. L'analyse de la complexité des approches proposées est discutée en section 6.5. Enfin, la section 6.6 conclut le chapitre.

## 6.2 Relation de dominance graduée

Pour réduire la taille du skyline, il est important de disposer d'une relation permettant de rendre ses éléments comparables, pour en sélectionner les meilleurs par rapport à la requête de l'utilisateur. Pour cela, nous proposons une relation de dominance graduée, appelée *f-dominance*, qui est plus sélective comparée à la relation de dominance de Pareto [3]. Elle est basée sur le quantificateur linguistique « *presque tous* », permettant ainsi d'identifier les points skyline qui répondent le mieux aux requêtes des utilisateurs *sur presque toutes les dimensions*, plutôt qu'au moins une seule dimension comme défini dans la relation de dominance de Pareto (voir la définition 2.3.1 du chapitre 2).

L'idée de base de notre approche de raffinement est de privilégier les points skyline qui satisfont le mieux la requête de l'utilisateur sur un maximum de critères parmi ses $d$ dimensions. À titre d'exemple, pour une requête skyline exprimée sur quatre attributs (ou dimensions), le point skyline $p_1\,(1,1,1,0)$ est bien meilleur que le point skyline $p_2\,(0,0,0,1)$, car $p_1$ répond le mieux à la requête de l'utilisateur que $p_2$ sur presque toutes les dimensions (i.e., trois attributs parmi quatre), où les valeurs 1 et 0 représentent respectivement la satisfaction totale et la non-satisfaction de la requête sur un critère donné. Pour cela, à chaque paire d'éléments du skyline, on associe un degré exprimant dans quelle mesure le premier élément est meilleur que le deuxième sur presque toutes les dimensions. À partir de ces degrés, on en déduit pour chaque point skyline en quelle mesure il est meilleur que tout les autres points du même skyline, ce qui permet de les ordonner des plus intéressants aux moins intéressants par rapport à la requête de l'utilisateur.

Considérons, dans ce qui suit, $A = \{a_1, ..., a_d\}$ un espace $d-$dimensionnel et $D = \{p_1, ..., p_m\}$ un ensemble de points (d'objets) $d$-dimensionnels sur $A$. La notation $p_i.a_r$ représente la valeur du point $p_i$ sur la $r^{ième}$ dimension. Par souci de simplicité et sans perte de généralité, nous supposons




(a) (b)

Figure 6.1: Exemples de définition du quantificateur linguistique « *presque tous* ».

que plus la valeur $p_i.a_r$ est grande, meilleure elle est.

La définition de la relation de dominance graduée utilisée pour raffiner un skyline de taille importante s'écrit :

**Définition 6.2.1 (Relation de f-dominance)**. Soit $S = \{p_1, ..., p_n\}$ le skyline sur $D$. On dit que le point skyline $p_i$ **f-domine** le point skyline $p_j$, noté $p_i \succ_{f-Dom} p_j$, si et seulement si $p_i$ est meilleur que $p_j$ **sur presque toutes les dimensions**.

Ainsi, un point skyline $p_i$ appartient au skyline raffiné $\mathbb{S}$ si et seulement si $p_i$ est meilleur que tous les autres points du même skyline sur **presque toutes les dimensions**. On peut observer à partir de la définition (6.2.1) que la relation de dominance graduée permet de définir dans quelle mesure un point skyline $p_i$ f-domine un autre point skyline $p_j$. Cette dominance graduelle s'appuie sur la fonction d'appartenance du quantificateur linguistique « *presque tous* ».

Des exemples de définition du quantificateur flou « *presque tous* » sont présentés dans la figure 6.1, où $p_i$ est un point intéressant du point de vue de l'utilisateur s'il répond au mieux à sa requête sur un maximum de dimensions (Figure 6.1a) ou bien sur au moins 80% des dimensions (Figure 6.1b). Dans cette dernière, l'utilisateur est totalement satisfait si et seulement si au moins 80% des attributs définis dans sa requête sont satisfaits.

À partir de cette définition, nous introduisons dans la section suivante, les méthodes de raffinement permettant de sélectionner les meilleurs points du skyline en prenant en considération le point de vue de l'utilisateur et la satisfaction des points skyline de chaque attribut présent dans la requête. Ceci est réalisé au moyen des outils qu'offre la théorie des ensembles flous pour évaluer dans quelle mesure un point skyline est meilleur qu'un autre point skyline par rapport à la requête de l'utilisateur.

## 6.3 Méthodes de raffinement

Dans cette section, nous proposons trois approches de raffinement de skyline fondées sur le quantificateur linguistique « *presque tous* » et la relation de dominance graduée définie dans la section

précédente [3]. Elles se basent également sur différents critères choisis par l'utilisateur pour discriminer les points skyline, permettant ainsi la sélection des meilleurs réponses vis-à-vis de l'utilisateur.

La première méthode (RDD), présentée en sous-section 6.3.1, permet de réduire la taille du skyline en utilisant l'information sur le nombre des dimensions dominantes lorsque l'on compare deux points skyline $p_i$ et $p_j$, i.e., le nombre des dimensions sur lesquelles $p_i$ est strictement meilleur que $p_j$. Quant à la deuxième approche (MRM) présentée en sous-section 6.3.2, elle considère à la fois le nombre des dimensions dominantes et les valeurs des deux points skyline comparés sur chaque dimension. Enfin, la troisième méthode de raffinement (RID) permet d'améliorer la discrimination des points skyline en considérant, en plus des deux critères utilisés par la deuxième approche, l'importance des dimensions (attributs) vis-à-vis de l'utilisateur.

### 6.3.1 Méthode de raffinement basée sur les dimensions dominantes (RDD)

Une façon basique et naturelle de discriminer deux points skyline $p_i$ et $p_j$ est tout simplement de considérer le nombre des dimensions dominantes entre ces deux points, c'est-à-dire, de calculer le nombre de dimensions sur lesquelles le point skyline $p_i$ est strictement meilleur que le point skyline $p_j$. Ensuite, nous évaluons le degré avec lequel $p_i$ *f-domine* le point $p_j$ en utilisant la fonction d'appartenance $\mu_{presTt}$ correspondant au quantificateur linguistique « *presque tous* » présentée dans la figure 6.1.

Cependant, pour ordonner les points skyline des plus satisfaisants aux moins satisfaisants par rapport à la requête de l'utilisateur, il est nécessaire d'évaluer dans quelle mesure un point skyline est meilleur *que tous les autres points skyline* suivant le critère lié au « nombre des dimensions dominantes », i.e., *sur presque toutes les dimensions*. Pour ce faire, la procédure à quatre étapes suivantes est appliquée :

Considérons, dans ce qui suit, $Q$ le quantificateur linguistique « *presque tous* » et $S = \{p_1, ..., p_n\}$ le skyline au sens de Pareto de $D$.

1. **Calcul du nombre des dimensions dominantes**

   Pour raffiner le skyline $S$, la première étape de notre approche RDD consiste à calculer le nombre des dimensions dominantes, noté $n_{ij}$, entre chaque deux points skyline $p_i$ et $p_j$, pour $i, j = 1, ..., n$ et $i \neq j$. Ceci permet donc de déterminer le nombre de dimensions sur lesquelles le point skyline $p_i$ répond le mieux à la requête de l'utilisateur par rapport au point skyline $p_j$. Le tableau 6.1 représente la matrice des nombres des dimensions dominantes entre les points du skyline $S$.

2. **Évaluation de la f-dominance entre deux points skyline**

   La deuxième étape, quant à elle, évalue le degré de satisfaction de la relation « *f-dominance* » suivant le nombre des dimensions dominantes entre chaque paire de points skyline ($p_i$,$p_j$), pour $i, j = 1, ..., n$ et $i \neq j$. Pour cela, le degré, noté $\mu_{f-Dom}[p_i, p_j]$, avec lequel $p_i$ *f-domine* $p_j$ est calculé comme suit :

TABLE 6.1 – Matrice des dimensions dominantes de $S$.

| | $p_1$ | $p_2$ | ... | $p_n$ |
|---|---|---|---|---|
| $p_1$ | - | $n_{12}$ | ... | $n_{1n}$ |
| $p_2$ | $n_{21}$ | - | | $n_{2n}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $p_n$ | $n_{n1}$ | $n_{n2}$ | ... | - |

$$\mu_{f-Dom}\left[p_i, p_j\right] = \mu_Q\left(\frac{n_{ij}}{d}\right) \tag{6.1}$$

où $\mu_Q$ est la fonction d'appartenance du quantificateur linguistique « *presque tous* » et $n_{ij}$ dénote le nombre des dimensions dominantes du point skyline $p_i$ par rapport au point skyline $p_j$. Par exemple, on peut prendre le quantificateur « presque tous » présenté dans la figure 6.1b.

Ainsi, la formule (6.1) détermine dans quelle mesure $p_i$ est strictement meilleur que $p_j$ sur presque toutes les dimensions.

3. **Évaluation globale de la f-dominance entre les points skyline**

   Pour ordonner les points skyline des plus intéressants aux moins intéressants suivant le nombre des dimensions satisfaisant le mieux la requête de l'utilisateur, le processus de raffinement de notre approche RDD calcule pour chaque point skyline $p_i$, son degré global, noté $\mu_{f-Dom}\left[p_i\right]$, avec lequel il *f-domine* tous les autres points skyline. Pour cela, pour chaque point skyline $p_i$, une fonction d'agrégation est appliquée sur l'ensemble de ses degrés élémentaires $\mu_{f-Dom}\left[p_i, p_j\right]$, évaluant le degré de f-dominance de $p_i$ par rapport à chaque point $p_j$ du skyline $S$, pour $j = 1, ..., n$ et $j \neq i$. Cette fonction peut être, par exemple, l'opérateur « min », la moyenne arithmétique, ...

   Ainsi, chaque point $p_i$ de cette nouvelle variante de skyline est associé à un degré exprimant dans quelle mesure $p_i$ *f-domine* tous les autres points skyline. Autrement dit, chaque élément $p_i$ du skyline $\mathbb{S}_{RDD}$ possède un degré global $\mu_{f-Dom}\left[p_i\right]$, déterminant à quel point « $p_i$ *est strictement meilleur que tous les autres points skyline sur presque toutes les dimensions* ».

4. **Ordonnancement et sélection des meilleurs points skyline**

   Enfin, les points skyline sont classés dans un ordre décroissant suivant leurs degrés de satisfaction de la f-dominance évalués dans l'étape précédente. Ainsi, pour réduire la taille du skyline, deux approches peuvent être appliquées :

   (a) l'approche basée sur un seuil $\sigma$, dans laquelle un point skyline $p_i$ est écarté de l'ensemble des meilleures réponses du skyline $S$, si est seulement si $\mu_{f-Dom}\left[p_i\right] < \sigma$, où $\sigma$ est un seuil défini par l'utilisateur ; ou bien

TABLE 6.2 – Un skyline $S$ sur un espace 8-dimensionnel $A$.

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | $a_8$ |
|---|---|---|---|---|---|---|---|---|
| $p_1$ | 0,6 | 0,4 | 0,6 | 0,7 | 0,2 | 0,5 | 0,4 | 0,3 |
| $p_2$ | 0,5 | 0,7 | 0,2 | 0,3 | 0,5 | 0,4 | 0,2 | 0,5 |
| $p_3$ | 0,7 | 0,2 | 0,5 | 0,9 | 0,5 | 0,7 | 0,3 | 0,3 |
| $p_4$ | 0,5 | 0,4 | 0,1 | 0,3 | 0,2 | 0,6 | 0,8 | 0,8 |
| $p_5$ | 0,8 | 0,7 | 0,5 | 0,6 | 0,5 | 0,2 | 0,2 | 0,2 |
| $p_6$ | 0,6 | 0,8 | 0,7 | 0,6 | 0,1 | 0,4 | 0,5 | 0,7 |

TABLE 6.3 – Matrice des dimensions dominantes des points skyline de $S$.

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ |
|---|---|---|---|---|---|---|
| $p_1$ | - | 5 | 3 | 3 | 5 | 3 |
| $p_2$ | 3 | - | 2 | 3 | 2 | 1 |
| $p_3$ | 4 | 6 | - | 5 | 4 | 4 |
| $p_4$ | 3 | 3 | 3 | - | 3 | 4 |
| $p_5$ | 3 | 3 | 2 | 5 | - | 2 |
| $p_6$ | 4 | 6 | 4 | 4 | 5 | - |

(b) la top-k approche permettant de sélectionner les $k$ meilleurs points skyline, où $k$ est le nombre de réponse attendues par l'utilisateur.

Dans l'algorithme 6.1 sont résumées les principales étapes de notre approche de raffinement RDD, où $\mu_{f-Dom}[p_i, p_j]$ représente le degré avec lequel le point skyline $p_i$ f-domine le point skyline $p_j$ et, *Agregation* dénote une fonction d'agrégation calculant le degré global, $\mu_{f-Dom}[p_i]$.

**Exemple 6.3.1.** Pour illustrer notre approche de raffinement RDD, considérons dans ce qui suit le skyline $S = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ résultant de l'évaluation d'une requête adressée à une base de données $D$ sur l'espace $A$ de 8 dimensions (i.e., $A = \{a_1, a_2, ..., a_8\}$). Voir le tableau 6.2.

Pour raffiner le skyline $S$ et sélectionner les meilleurs de ses points par rapport à la requête de l'utilisateur, nous appliquons l'algorithme 6.1 comme suit :

Dans un premier temps, nous calculons à partir des résultats du tableau 6.2, le nombre des dimensions dominantes $n_{ij}$ de chaque point skyline $p_i$ comparé à un autre point skyline $p_j$, pour $i, j = 1, ..., n$ et $i \neq j$. La matrice regroupant les nombres des dimensions dominantes des points skyline de $S$ est illustrée dans le tableau 6.3. À titre d'exemple, le point $p_2$ est strictement meilleur que $p_6$ seulement sur la $5^e$ dimension tandis que $p_6$ est strictement meilleur que $p_2$ sur six dimensions.

Par la suite, le degré $\mu_{f-Dom}[p_i, p_j]$ avec lequel un point skyline $p_i$ *f-domine* un autre point $p_j$ du même skyline est évalué suivant la fonction d'appartenance du quantificateur flou « *presque tous* » présentée dans la figure 6.1a, où le nombre de dimensions $d = 8$. Le tableau 6.4 présente l'ensemble des degrés de satisfaction de la f-dominance entre chaque deux points skyline comparés. Ainsi, on peut dire que le point $p_2$ *f-domine* $p_6$ avec un degré égal à 0,125 et que $p_6$ *f-domine* $p_2$ avec un degré

**Algorithme 6.1** RDD.

**Entrée :** Un skyline $S = \{p_1, ..., p_n\}$ d'un ensemble de points $D$ sur l'espace $A = \{a_1, ..., a_d\}$.
**Sortie :** Un skyline raffiné $\mathbb{S}_{RDD}$.
**Début**

// Initialisation du skyline raffiné $\mathbb{S}_{RDD}$ à l'ensemble vide
$\mathbb{S}_{RDD} \leftarrow \emptyset$
// Évaluation du nombre des dimensions dominantes
// entre chaque paire de points skyline
**Pour chaque** paire de points skyline $(p_i, p_j) \in S^2$ et $i \neq j$ **faire**
  **Pour** $r$ allant de 1 à $d$ **faire** calculer $n_{ij}$
// Calcul du degré de la f-dominance entre chaque deux points skyline
**Pour chaque** paire de points skyline $(p_i, p_j) \in S^2$ et $i \neq j$ **faire**
  $\mu_{f-Dom}[p_i, p_j] \leftarrow \mu_Q\left(\frac{n_{ij}}{d}\right)$
// Évaluation globale de la f-dominance de chaque point skyline
**Pour chaque** point skyline $p_i \in S$ **faire**
  $\mu_{f-Dom}[p_i] \leftarrow \underset{1 \leq j \leq n \text{ et } j \neq i}{Agregation}\left(\mu_{f-Dom}[p_i, p_j]\right)$
// Ordonnancement des points skyline
**Pour** $i$ allant de 1 à $n$ **faire**
  classer dans un ordre décroissant les points skyline $p_i$
  suivant leurs degrés $\mu_{f-Dom}[p_i]$
// Sélection des meilleurs points skyline
Soit *select* la méthode de sélection des points skyline de $S$
**Si** *select* = "$a$" **alors**
  // Méthode de sélection basée sur un seuil $\sigma$ // avec $\sigma$ un seuil défini par l'utilisateur
  **Pour** chaque point skyline $p_i \in S$ **faire**
    **Si** $\mu_{f-Dom}[p_i] \geq \sigma$ **alors**
      ajouter $p_i$ à l'ensemble $\mathbb{S}_{RDD}$
      **Sinon** écarter $p_i$
    **Fin si** ;
  **Fin pour** ;
  **Sinon Si** *select* = "$b$" **alors**
    // Sélection suivant la top-k approche // avec $k$ un paramètre introduit par l'utilisateur
    sélectionner les top-k points skyline
    ajouter les $k$ points skyline sélectionnés à $\mathbb{S}_{RDD}$
  **Fin si** ;
**Fin si** ;

Retourner $\mathbb{S}_{RDD}$

**Fin**

TABLE 6.4 – Les degrés de f-dominance, $\mu_{f-Dom}[p_i, p_j]$, entre les points du skyline $S$.

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $\mu_{f-Dom}[p_i]$ |
|---|---|---|---|---|---|---|---|
| $p_1$ | - | 0,625 | 0,375 | 0,375 | 0,625 | 0,375 | **0,375** |
| $p_2$ | 0,375 | - | 0,25 | 0,375 | 0,25 | 0,125 | **0,125** |
| $p_3$ | 0,5 | 0,75 | - | 0,625 | 0,5 | 0,5 | **0,5** |
| $p_4$ | 0,375 | 0,375 | 0,375 | - | 0,375 | 0,5 | **0,375** |
| $p_5$ | 0,375 | 0,375 | 0,25 | 0,625 | - | 0,25 | **0,25** |
| $p_6$ | 0,5 | 0,75 | 0,5 | 0,5 | 0,625 | - | **0,5** |

égal à 0,75. Ceci peut être interprété comme suit : « $p_2$ *est strictement meilleur que* $p_6$ *sur au moins 12,5% des dimensions, et* $p_6$ *est strictement meilleur que* $p_2$ *sur au moins 75% des dimensions* ».

Pour évaluer le degré global $\mu_{f-Dom}[p_i]$ avec lequel chaque point skyline $p_i$ *f-domine tous les autres points du skyline* $S$, nous appliquons l'opérateur « min » comme fonction d'agrégation (voir la dernière colonne du tableau 6.4). Ainsi, une variante graduelle du skyline $S$, notée $\mathbb{S}_{RDD}$, est obtenue et peut s'écrire comme suit :

$\mathbb{S}_{RDD} = \{(0,375)/p_1, (0,125)/p_2, (0,5)/p_3, (0,375)/p_4, (0,25)/p_5, (0,5)/p_6\}$.

Enfin, le skyline raffiné $\mathbb{S}_{RDD}$ suivant le nombre des dimensions dominantes est :

- $\mathbb{S}_{RDD}(k)$ contenant les top-k points skyline de $S$, par exemple, $\mathbb{S}_{RDD}(3) = \{p_3, p_6, p_1$ ou bien $p_4\}$ ; ou bien
- $\mathbb{S}_{RDD}(\sigma) = \{p \in S | \mu_{f-Dom}[p] \geq \sigma\}$, par exemple, $\mathbb{S}_{RDD}(0,5) = \{p_3, p_6\}$. □

En examinant le résultat retourné par notre approche RDD, nous constatons que malgré la différence entre les valeurs de $p_1$ et de $p_4$ sur chaque dimension, ces derniers f-dominent tous les autres points skyline avec le même degré global 0,375 (i.e., $p_1$ et $p_4$ sont meilleurs que tous les autres points skyline sur au moins 4 dimensions), ce qui signifie que l'approche RDD peut conduire à une faible discrimination des points skyline car elle ne prend en compte que le nombre des dimensions dominantes lorsque l'on compare deux points skyline.

Pour cela, nous proposons dans la sous-section suivante une approche complémentaire permettant de mieux discriminer les points d'un même skyline, en prenant en compte les valeurs des attributs.

### 6.3.2 Méthode de raffinement mixte (MRM)

Afin d'améliorer le critère de discrimination des points skyline de l'approche RDD, nous proposons dans cette sous-section une méthode de raffinement de skyline complémentaire à cette dernière, en prenant en considération non seulement le nombre des dimensions dominantes mais aussi la mesure dans laquelle un point skyline est meilleur qu'un autre point sur chaque dimension.

Cette nouvelle approche de raffinement, notée MRM, se base sur la définition (6.2.1) de la section 6.2 et évalue la mesure dans laquelle « *un point skyline* $p_i$ *f-domine un autre point skyline* $p_j$ » en calculant le degré de vérité de la proposition suivante :

$\gamma'_1$ : presque toutes les dimensions de $p_i$ sont meilleures que celles de $p_j$.

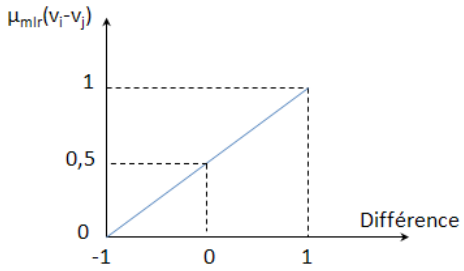



FIGURE 6.2 – Fonction d'appartenance $\mu_{mlr}$ du prédicat flou « *meilleur* ».

La proposition ci-dessus est une expression quantifiée floue de la forme « $Q$ $X$ sont $P$ », où (i) $Q$ est un quantificateur relatif « *presque tous* » [61] qui est défini par une fonction d'appartenance $\mu_Q$ telle que $\mu_Q(\varpi)$ est le degré de vérité de « $Q$ $X$ sont $P$ » quand une proportion $\varpi$ d'éléments de $X$ satisfait complètement $P$ et les autres éléments n'étant pas satisfaits ; (ii) $X$ est un ensemble d'éléments ; (iii) $P$ est un prédicat flou.

Pour évaluer le degré de vérité de la proposition $\gamma_1'$, noté $\mu_{f-Dom}\left[p_i, p_j\right]$, nous appliquons la méthode par décomposition proposée dans [142] (voir la section 2.2 du chapitre 2), en considérant $X$ l'ensemble des dimensions des points skyline (i.e., $|X| = d$) et $\mu_Q$ la fonction d'appartenance du quantificateur relatif « *presque tous* ». La sémantique de ce dernier est présentée dans la figure 6.1a. Ainsi, le degré de vérité $\mu_{f-Dom}\left[p_i, p_j\right]$ de la proposition $\gamma_1'$ est calculé comme suit :

$$\mu_{f-Dom}\left[p_i, p_j\right] = \max_{1 \leq r \leq d} \min\left(\mu_r, \mu_Q\left(r/d\right)\right) \tag{6.2}$$

où $\Omega = \{\mu_1 : \mu_{mlr}\left(f\left(p_i.a_1, p_j.a_1\right)\right), ..., \mu_d : \mu_{mlr}\left(f\left(p_i.a_d, p_j.a_d\right)\right)\}$ est l'ensemble des degrés (donnés par $\mu_{mlr}$) avec lesquels $p_i \in S$ est *plus ou moins meilleur* que $p_j \in S$ sur la dimension $a_r$ $(1 \leq r \leq d)$ en prenant en considération les deux valeurs $p_i.a_r$ et $p_j.a_r$ tels que $\mu_1 \geq \ldots \geq \mu_d$, $\mu_Q$ dénote la fonction d'appartenance du quantificateur linguistique « presque tous » et, $f$ est une fonction évaluant de combien une valeur $p_i.a_r$ est strictement meilleure qu'une autre valeur $p_j.a_r$. Dans le cas numérique, cette quantité (i.e., $f\left(p_i.a_r, p_j.a_r\right)$) représente la différence entre les deux valeurs $v_i = p_i.a_r$ et $v_j = p_j.a_r$. La fonction d'appartenance $\mu_{mlr}$ du prédicat flou « *meilleur* » peut être représentée par une fonction linéaire sur l'intervalle $[-1, 1]$ comme illustrée dans la figure 6.2. Dans le cas où la différence est négative (resp., positive), $\mu_{mlr}$ permet d'évaluer dans quelle mesure le point $p_i$ est moins (resp., plus) intéressant par rapport au point $p_j$.

Par souci de simplicité et sans perte de généralité, nous considérons dans la suite de ce chapitre que les deux fonctions $\mu_{mlr}$ et $f$ sont les mêmes dans toutes les dimensions lors de l'évaluation des points skyline. Néanmoins, elles peuvent être adaptées aux cas où la propriété de commensurabilité n'est pas vérifiée, en définissant deux fonctions, $\mu_{mlr}^r$ et $f^r$, spécifiques à chaque dimension $a_r$, pour $r = 1, ..., d$.

En résumé, la fonction $\mu_Q$ évalue le degré avec lequel un point $p_i \in S$ *f-domine* un autre point $p_j \in S$ suivant le nombre des dimensions dominantes $n_{ij}$. Quant à la fonction $\mu_{mlr}$, elle calcule la mesure avec laquelle $p_i \in S$ *f-domine* $p_j \in S$ en prenant en compte à quel point $p_i$ est meilleur que

$p_j$ sur chaque dimension $a_r$ ($r = 1, ..., d$). Enfin, le degré de vérité $\mu_{f-Dom}[p_i, p_j]$ exprime à quel point $p_i$ f-domine $p_j$ en tenant compte (i) de toutes les dimensions et (ii) des informations liées aux deux valeurs $\mu_{mlr}$ et $\mu_Q$ correspondantes.

Dans l'objectif de retourner aux utilisateurs des degrés de vérité ayant une sémantique claire [96], nous considérons en particulier la formule (6.2) dans le cas où $\mu_Q(r/d) = r/d$. De cette façon, le degré de vérité $\mu_{f-Dom}[p_i, p_j] = \alpha$ de la proposition $\gamma'_1$ signifie que « au moins $\alpha'$% des dimensions de $p_i$ sont strictement meilleures que celles de $p_j$ à au moins un degré de $\alpha$, où $\alpha' = 100 \times \alpha$.

Pour extraire et ordonner les meilleurs points du skyline $S$ selon l'approche de raffinement MRM, les étapes suivantes sont donc appliquées :

Considérons, dans ce qui suit, $\mu_Q$ la fonction d'appartenance du quantificateur linguistique « *presque tous* », $S = \{p_1, ..., p_n\}$ le skyline de $D$ et $\mu_{mlr}$ la fonction d'appartenance du prédicat « *meilleur* ».

1. **Détermination des quantités** $f(p_i.a_r, p_j.a_r)$

   La première étape pour réduire la taille du skyline $S$ consiste à calculer pour chaque paire de points skyline $(p_i, p_j)$, pour $i, j = 1, ..., n$ et $i \neq j$, la quantité avec laquelle $p_i$ est meilleur que $p_j$ sur chaque dimension $a_r$.

2. **Évaluation des degrés** $\mu_{mlr}(f(p_i.a_r, p_j.a_r))$

   À partir des valeurs $f(p_i.a_r, p_j.a_r)$ calculées dans l'étape précédente (pour $r = 1, ..., d$ ; $i, j = 1, ..., n$ ; $i \neq j$), les degrés $\mu_{mlr}(f(p_i.a_r, p_j.a_r))$ représentant la satisfaction du prédicat « *meilleur* » lorsque l'on compare deux points skyline sur une dimension $a_r$, sont évalués. Ceci permet donc de déterminer à quel degré le point $p_i \in S$ est plus ou moins meilleur que $p_j \in S$ sur chaque dimension $a_r$, avec $r = 1, ..., d$.

3. **Calcul des degrés de vérité** $\mu_{f-Dom}[p_i, p_j]$

   Pour prendre en considération les deux critères de discrimination, à savoir, le nombre des dimensions dominantes et la valeur de chaque dimension lorsque l'on compare deux points skyline $p_i$ et $p_j$, le processus de raffinement de la méthode MRM applique, en troisième étape, la formule (6.2) permettant de calculer le degré de vérité, $\mu_{f-Dom}[p_i, p_j]$, de l'expression « *presque toutes les dimensions de* $p_i$ *sont strictement meilleures que celles de* $p_j$ », pour $i, j = 1, ..., n$ et $i \neq j$.

4. **Évaluation des degrés de f-dominance** $\mu_{f-Dom}[p_i]$

   Pour déterminer dans quelle mesure un point skyline $p_i$ est plus intéressant par rapport *à tous les autres points du même skyline* vis-à-vis de l'utilisateur, le processus de raffinement MRM calcule son degré global $\mu_{f-Dom}[p_i]$ au moyen d'une fonction d'agrégation. Cette dernière combine les degrés de vérité élémentaires $\mu_{f-Dom}[p_i, p_j]$ de $p_i$ en utilisant, par exemple, la moyenne arithmétique, l'opérateur « min », ...

TABLE 6.5 – Les quantités $f(p_i.a_r, p_j.a_r)$, pour $i, j = 1, ..., 6$, $r = 1, ..., 8$ et $i \neq j$.

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | $a_8$ |
|---|---|---|---|---|---|---|---|---|
| $f(p_1, p_2)$ | 0,1 | -0,3 | 0,4 | 0,4 | -0,3 | 0,1 | 0,2 | -0,2 |
| $f(p_1, p_3)$ | -0,1 | 0,2 | 0,1 | -0,2 | -0,3 | -0,2 | 0,1 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $f(p_4, p_6)$ | -0,1 | -0,4 | -0,6 | -0,3 | 0,1 | 0,2 | 0,3 | 0,1 |
| $f(p_5, p_6)$ | 0,2 | -0,1 | -0,2 | 0 | 0,4 | -0,2 | -0,3 | -0,5 |

Autrement dit, le processus de raffinement MRM associe à chaque point skyline $p_i$ un degré global $\mu_{f-Dom}[p_i]$ exprimant dans quelle mesure « $p_i$ *est strictement meilleur que tous les autres points skyline sur presque toutes les dimensions* ».

5. **Ordonnancement et sélection des meilleurs points skyline**

Enfin, la dernière étape consiste à classer les points skyline $p_i$, pour $i = 1, ..., n$, dans un ordre décroissant suivant leurs degrés de satisfaction de la f-dominance, $\mu_{f-Dom}[p_i]$. Ainsi, pour réduire la taille du skyline, les approches suivantes peuvent être appliquées :

(a) l'approche basée sur un seuil $\sigma$, dans laquelle un point skyline $p_i$ est écarté de l'ensemble des meilleures réponses du skyline $S$, si est seulement si $\mu_{f-Dom}[p_i] < \sigma$, où $\sigma$ est un seuil défini par l'utilisateur ; ou bien

(b) la top-k approche permettant de sélectionner les $k$ meilleurs points skyline, où $k$ est le nombre de réponses attendues par l'utilisateur.

L'algorithme 6.2 résume les principales étapes de la procédure de raffinement MRM.

**Exemple 6.3.2.** Pour raffiner le skyline $S = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ de l'exemple 6.3.1 suivant la définition (6.2.1) de la f-dominance et de l'approche de raffinement MRM décrite dans l'algorithme 6.2, nous procédons comme suit :

À partir des valeurs du tableau 6.2, nous calculons la différence $f(p_i.a_r, p_j.a_r)$ entre les valeurs $p_i.a_r$ et $p_j.a_r$, dénotant de combien le point skyline $p_i$ est strictement meilleur que le point skyline $p_j$ sur la dimension $a_r$. Un extrait de ces quantités est présenté dans le tableau 6.5.

Par la suite, les degrés $\mu_{mlr}(f(p_i.a_r, p_j.a_r))$ entre chaque deux points skyline $p_i$ et $p_j$ sont calculés, en appliquant la fonction d'appartenance du prédicat « *meilleur* » illustrée dans la figure 6.2, sur les valeurs obtenues dans l'étape précédente (i.e., $f(p_i.a_r, p_j.a_r)$). Chacun de ces degrés exprime donc la satisfaction du prédicat « *meilleur* » lorsque l'on compare deux points skyline. Un extrait de ces degrés de satisfaction est présenté dans le tableau 6.6. Par exemple, les degrés avec lesquels le point $p_1$ est plus ou moins meilleur que $p_2$ sont : $\mu_{mlr}(f(p_1.a_1, p_2.a_1)) = \mu_{mlr}(f(p_1.a_6, p_2.a_6)) = 0,55$ ; $\mu_{mlr}(f(p_1.a_2, p_2.a_2)) = \mu_{mlr}(f(p_1.a_5, p_2.a_5)) = 0,35$ ; $\mu_{mlr}(f(p_1.a_3, p_2.a_3)) = \mu_{mlr}(f(p_1.a_4, p_2.a_4)) = 0,7$ ; $\mu_{mlr}(f(p_1.a_7, p_2.a_7)) = 0,6$ et $\mu_{mlr}(f(p_1.a_8, p_2.a_8)) = 0,4$.

Ensuite, nous calculons le degré de vérité de la f-dominance, $\mu_{f-Dom}[p_i, p_j]$, de chaque paire $(p_i, p_j)$ de points skyline. Pour cela, nous appliquons la formule (6.2) basée sur la satisfaction du

**Algorithme 6.2** MRM.

**Entrée :** Un skyline $S = \{p_1, ..., p_n\}$ d'un ensemble de points $D$ sur l'espace $A = \{a_1, ..., a_d\}$.
**Sortie :** Un skyline raffiné $\mathbb{S}_{MRM}$.
**Début**

// Initialisation du skyline raffiné $\mathbb{S}_{MRM}$ à l'ensemble vide
$\mathbb{S}_{MRM} \leftarrow \emptyset$
// Calcul des quantités $f(p_i.a_r, p_j.a_r)$
// entre chaque paire de points skyline sur chaque dimension
**Pour chaque** paire de points skyline $(p_i, p_j) \in S^2$ et $j = i + 1, ..., n$ **faire**
  **Pour** $r$ allant de 1 à $d$ **faire** calculer $f(p_i.a_r, p_j.a_r)$
// Évaluation des degrés $\mu_{mlr}(f(p_i.a_r, p_j.a_r))$
// pour mesurer la satisfaction du prédicat « *meilleur* » sur chaque dimension
**Pour chaque** paire de points skyline $(p_i, p_j) \in S^2$ et $i \neq j$ **faire**
**Pour** $r$ allant de 1 à $d$ **faire** évaluer $\mu_{mlr}(f(p_i.a_r, p_j.a_r))$
// Calcul du degré $\mu_{f-Dom}[p_i, p_j]$
// pour mesurer le degré de dominance entre chaque deux points skyline
**Pour chaque** paire de points skyline $(p_i, p_j) \in S^2$ et $i \neq j$ **faire**
  $\mu_{f-Dom}[p_i, p_j] \leftarrow \max_{1 \leq r \leq d} \min(\mu_r, \mu_Q(r/d))$
// Évaluation globale de la f-dominance de chaque point skyline
**Pour chaque** point skyline $p_i \in S$ **faire**
  $\mu_{f-Dom}[p_i] \leftarrow \underset{1 \leq j \leq n \text{ et } j \neq i}{Agregation}(\mu_{f-Dom}[p_i, p_j])$
// Ordonnancement des points skyline
**Pour** $i$ allant de 1 à $n$ **faire**
  classer dans un ordre décroissant les points skyline $p_i$
  suivant leurs degrés $\mu_{f-Dom}[p_i]$
// Sélection des meilleurs points skyline
Soit *select* la méthode de sélection des points skyline de $S$
**Si** *select* = "$a$" **alors**
  // Méthode de sélection basée sur un seuil $\sigma$ // avec $\sigma$ un seuil défini par l'utilisateur
  **Pour** chaque point skyline $p_i \in S$ **faire**
    **Si** $\mu_{f-Dom}[p_i] \geq \sigma$ **alors** ajouter $p_i$ à l'ensemble $\mathbb{S}_{MRM}$
      **Sinon** écarter $p_i$
    **Fin si** ;
  **Fin pour** ;
  **Sinon Si** *select* = "$b$" **alors**
    // Sélection suivant la top-k approche // avec $k$ un paramètre introduit par l'utilisateur
    sélectionner les top-k points skyline
    ajouter les $k$ points skyline sélectionnés à $\mathbb{S}_{MRM}$
  **Fin si** ;
**Fin si** ;

Retourner $\mathbb{S}_{MRM}$

**Fin**

TABLE 6.6 – Les degrés de satisfaction, $\mu_{mlr}$, du prédicat flou « *meilleur* ».

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | $a_8$ |
|---|---|---|---|---|---|---|---|---|
| $\mu_{mlr}(f(p_1,p_2))$ | 0,55 | 0,35 | 0,7 | 0,7 | 0,35 | 0,55 | 0,6 | 0,4 |
| $\mu_{mlr}(f(p_1,p_3))$ | 0,45 | 0,6 | 0,55 | 0,4 | 0,35 | 0,4 | 0,55 | 0,5 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $\mu_{mlr}(f(p_4,p_6))$ | 0,45 | 0,3 | 0,2 | 0,35 | 0,55 | 0,6 | 0,65 | 0,55 |
| $\mu_{mlr}(f(p_5,p_6))$ | 0,6 | 0,45 | 0,4 | 0,5 | 0,7 | 0,4 | 0,35 | 0,25 |

TABLE 6.7 – Matrice des degrés de f-dominance, $\mu_{f-Dom}[p_i]$, de chaque point skyline $p_i$.

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $\mu_{f-Dom}[p_i]$ |
|---|---|---|---|---|---|---|---|
| $p_1$ | - | 0,55 | 0,5 | 0,5 | 0,55 | 0,5 | **0,5** |
| $p_2$ | 0,45 | - | 0,45 | 0,5 | 0,5 | 0,45 | **0,45** |
| $p_3$ | 0,5 | 0,55 | - | 0,55 | 0,5 | 0,5 | **0,5** |
| $p_4$ | 0,5 | 0,5 | 0,45 | - | 0,375 | 0,5 | **0,375** |
| $p_5$ | 0,45 | 0,5 | 0,5 | 0,625 | - | 0,45 | **0,45** |
| $p_6$ | 0,5 | 0,55 | 0,5 | 0,5 | 0,55 | - | **0,5** |

prédicat « *meilleur* » ainsi que sur celle du quantificateur linguistique « *presque tous* ». L'ensemble des degrés de f-dominance des points skyline est illustré dans le tableau 6.7. À titre d'exemple, le degré avec lequel le point skyline $p_1$ est strictement meilleur que (i.e., *f-domine*) $p_2$ sur presque toutes les dimensions est évalué comme suit :

$\mu_{f-Dom}[p_1,p_2] = \max_{a_{(r)};1\leq r\leq 8} \min(\mu_{mlr}(f(p_1.a_{(r)}, p_2.a_{(r)})), \mu_{presTt}(r/8))$ ;

$\mu_{f-Dom}[p_1,p_2] = \max(\min(0.70, 1/8), \ldots, \min(0.35, 8/8))$.

Ainsi, le degré de f-dominance de $p_1$ comparé à $p_2$ est égal à 0.55. En d'autres termes, ce degré signifie qu'*au moins 55% des dimensions de $p_1$ sont strictement meilleures que celles de $p_2$ à au moins un degré de 0,55.*

Enfin, pour sélectionner les points skyline les plus satisfaisants, nous évaluons le degré global de f-dominance, $\mu_{f-Dom}[p_i]$, de chaque point skyline $p_i$ par rapport à tous les autres points du skyline $S$. Pour cela, nous utilisons, comme dans l'exemple 6.3.1., l'opérateur « min » comme fonction d'agrégation entre les degrés de vérité $\mu_{f-Dom}[p_i,p_j]$, pour $j = 1, ..., 6$ et $j \neq i$. Les résultats obtenus sont illustrés dans la dernière colonne du tableau 6.7.

Par conséquent, l'ensemble des top-k(= 3) points skyline de $S$ suivant notre approche MRM est $\mathbb{S}_{MRM}(3) = \{(0,5)/p_1, (0,5)/p_3, (0,5)/p_6\}$. Ainsi, contrairement à l'approche RDD appliquée dans l'exemple 6.3.1, l'algorithme 6.2 a réussi à différencier entre les deux points skyline $p_1$ et $p_4$ en déterminant lequel d'entre eux est le plus intéressant par rapport à la requête de l'utilisateur. □

En conclusion, comme attendu, l'approche MRM permet d'améliorer la discrimination des points skyline comparée à l'approche RDD définie dans la sous-section précédente. Toutefois, elle peut retourner plusieurs points skyline avec le même degré de f-dominance, tel est le cas pour les points $p_1$, $p_3$ et $p_6$ qui ont un degré égal à 0,5.

Pour pallier cet inconvénient, nous proposons dans la sous-section suivante une troisième approche permettant de discriminer encore plus les points skyline en considérant -en plus des deux critères pris en compte par l'approche MRM- l'importance de chaque dimension du point de vue de l'utilisateur.

### 6.3.3 Méthode de raffinement basée sur l'importance des dimensions (RID)

Nous présentons, dans cette sous-section, une approche de raffinement de skyline plus discriminante, permettant cette fois-ci de prendre en considération l'ensemble des dimensions importantes $X' \subset A$, en plus des critères de l'approche MRM, à savoir, (i) le nombre des dimensions dominantes et (ii) la valeur de chaque point skyline sur chaque dimension importante.

À partir de la définition (6.2.1), nous considérons qu'un point skyline $p_i$ appartient au skyline raffiné $\mathbb{S}_{RID}$ suivant l'ensemble $X'$ des dimensions importantes (au lieu de toutes les dimensions de l'espace $A$), si et seulement si « $p_i$ ***est strictement meilleur que tous les autres points skyline sur presque toutes les dimensions importantes*** ». Ainsi, le degré de vérité de la proposition suivante peut être évalué lorsque l'on compare deux points skyline $p_i$ et $p_j$ :

$\gamma'_2$ : presque toutes les dimensions importantes de $p_i$ sont meilleures que celles de $p_j$.

Cette dernière est une proposition quantifiée floue de la forme « $Q$ $B$ $X$ sont $P$ », où $B$ et $P$ sont deux prédicats flous. Cependant, ce type de proposition est difficile à interpréter lorsque l'ensemble des éléments vérifiant le prédicat « $B$ » est vide (ou presque vide). Pour remédier à cet inconvénient, nous considérons dans cette approche un prédicat « $B$ » de type booléen, permettant ainsi de transformer la proposition précédente en « $Q$ $X'$ sont $P$ », où $X'$ représente l'ensemble des dimensions importantes définies par l'utilisateur. Voir la sous-section 2.2.2 du chapitre 2.

De cette façon, la méthode par décomposition de Yager [142] peut être utilisée pour évaluer le degré de vérité $\mu_{f-Dom}[p_i, p_j]$ de la proposition $\gamma'_2$. Ce degré est donc calculé par la formule suivante :

$$\mu_{f-Dom}[p_i, p_j] = \max_{1 \leq r \leq |X'|} \min\left(\mu_r, \mu_Q\left(r/|X'|\right)\right) \tag{6.3}$$

où $\Omega = \{\mu_1 : \mu_{mlr}\left(f\left(p_i.a_1, p_j.a_1\right)\right), ..., \mu_d : \mu_{mlr}\left(f\left(p_i.a_d, p_j.a_d\right)\right)\}$ est l'ensemble des degrés de satisfaction $\mu_{mlr}$ avec lesquels $p_i \in S$ est *plus ou moins meilleur* que $p_j \in S$ sur chaque dimension importante $a_r$ ($a_r \in X'$).

Ainsi, la procédure de raffinement de skyline selon l'approche RID est une variante de l'algorithme 6.2 de la méthode MRM en substituant le nombre de dimensions « $d$ » par celui des dimensions importantes, i.e., « $|X'|$ ».

L'approche RID discrimine les points skyline et permet d'identifier les plus intéressants en considérant à la fois : (i) les dimensions les plus importantes vis-à-vis de l'utilisateur, (ii) leurs valeurs et (iii) le nombre des dimensions dominantes parmi les plus importantes.

**Exemple 6.3.3.** Reprenons l'exemple 6.3.2 de la section précédente où $\mathbb{S}_{MRM}(3) = \{(0,5)/p_1, (0,5)/p_3, (0,5)/p_6\}$. Considérons $X' = \{a_2, a_4, a_6\}$ l'ensemble des dimensions les plus importantes du

point de vue de l'utilisateur. Pour mieux discriminer les points skyline $p_1$, $p_3$ et $p_6$, nous appliquons la méthode RID comme suit :

Comme les valeurs $f(p_i.a_r, p_j.a_r)$ et $\mu_{mlr} f(p_i.a_r, p_j.a_r)$ entre les points skyline $p_1$, $p_3$ et $p_6$ sont déjà calculées dans l'exemple 6.3.2, nous évaluons directement leurs degrés de vérité $\mu_{f-Dom}[p_i, p_j]$ en utilisant la formule (6.3), pour $i, j \in \{1, 3, 6\}$ et $i \neq j$. Ensuite, l'opérateur « min » est appliqué pour déterminer leur degré global $\mu_{f-Dom}[p_i]$, c'est-à-dire : $\mu_{f-Dom}[p_1] = 0,55$, $\mu_{f-Dom}[p_3] = 0,4$ et $\mu_{f-Dom}[p_6] = 0,35$.

Par conséquent, en prenant en considération l'ensemble des dimensions importantes en plus des deux critères de discrimination utilisés dans l'approche MRM, l'ensemble des top-k(= 2) points du skyline $S$ est $\mathbb{S}_{RID}(2) = \{p_1, p_3\}$. □

## 6.4 Méthodologie de raffinement

Afin de raffiner un skyline ayant un grand nombre de réponses, nous avons présenté dans la section précédente trois méthodes de raffinement pour sélectionner les meilleurs points skyline suivant des critères définis par l'utilisateur. Ces critères sont au nombre de trois : (i) le nombre des dimensions dominantes, (ii) les valeurs des points skyline sur chaque dimension et (iii) l'ensemble des dimensions les plus importantes du point de vue de l'utilisateur.

La première approche, appelée RDD, permet de réduire un skyline en utilisant la notion de dimensions dominantes, permettant ainsi de préférer un point skyline $p_i$ à un autre point $p_j$ du même skyline si celui-ci est strictement meilleur que $p_j$ sur un plus grand nombre de dimensions. La deuxième méthode MRM est appliquée lorsque les deux critères sélectionnés pour discriminer les points skyline sont : (i) le nombre des dimensions dominantes et (ii) la valeur de chaque point skyline sur chaque dimension. Enfin, la troisième approche RID est très similaire à l'approche MRM mais qui applique les deux critères de discrimination cités ci-dessus sur l'ensemble des dimensions les plus importantes vis-à-vis de l'utilisateur.

De cette façon, on peut dire que ces trois méthodes peuvent être complémentaires dans le sens où (i) l'approche MRM permet de discriminer les points skyline lorsque l'utilisateur souhaite raffiner encore plus les réponses retournées par la méthode RDD, et (ii) la troisième approche, dite RID, permet également de rendre les points skyline plus discriminants en comparant leurs valeurs sur les dimensions importantes lorsque l'approche MRM n'est pas suffisante du point de vue de l'utilisateur.

Pour cela, nous décrivons dans ce qui suit une méthodologie pour raffiner un skyline de taille importante. Elle peut être intégrée au niveau d'un système d'interrogation de bases de données multidimensionnelles, pour aider à la réalisation du raffinement de skyline. Cette méthodologie consiste en cinq étapes comme illustrée dans la figure 6.3. Ces étapes sont appliquées dans l'ordre qui suit :

1. **Détermination du skyline à raffiner :** cette étape consiste à déterminer l'ensemble des points skyline à raffiner pour identifier les plus intéressants du point de vue de l'utilisateur. Cet ensemble peut être soit un skyline d'une base de données multidimensionnelles ou bien un

skyline raffiné obtenu par la méthode MRM.

2. **Sélection des critères de discrimination :** cette étape vise à identifier les critères permettant de discriminer les points skyline pour appliquer par la suite l'approche de raffinement correspondante (i.e., RDD, MRM ou bien RID).

3. **Identification de la méthode de sélection :** Cette étape consiste à sélectionner soit (i) l'approche basée sur le seuil $\sigma$ permettant de choisir tous les points skyline qui sont strictement meilleurs que tous les autres points du même skyline sur presque toutes les dimensions à au moins un degré égal à $\sigma$ ; ou bien (ii) la top-k approche permettant de retourner l'ensemble des $k$ meilleurs points skyline par rapport à la requête de l'utilisateur.

4. **Application de l'approche de raffinement correspondante :** cette étape correspond à l'exécution de l'approche de raffinement correspondant aux critères de discrimination choisis dans l'étape 2. Pour chaque ensemble de critères identifiés, nous appliquons l'approche de raffinement qui lui est associée pour réduire la taille du skyline et en choisir ses meilleurs points vis-à-vis de l'utilisateur. Par exemple, pour appliquer l'approche de raffinement MRM, les critères (i) nombre des dimensions dominantes et (ii) les valeurs des dimensions, doivent être sélectionnés.

5. **Présentation des résultats :** cette étape consiste à valider ou pas les résultats obtenus. Deux cas sont possibles :

   5.1. **Sauvegarde des résultats :** si les résultats obtenus lors du raffinement sont validés par l'utilisateur alors ils seront soit utilisés pour une évaluation comparative avec les résultats des autres approches de raffinement ou bien seront sauvegardés pour une future réutilisation ou encore utilisés comme données d'entrée d'une autre approche de raffinement.

   5.2. **Modification des critères de discrimination :** si les résultats ne sont pas validés, ce qui correspond par exemple à une faible discrimination des points skyline (i.e., plusieurs points skyline ont le même degré de f-dominance), alors il faudrait modifier les critères de discrimination et appliquer une autre méthode de raffinement plus sélective.

En résumé, la première étape porte sur les points skyline pour réaliser le raffinement. Les deux étapes suivantes (2 et 3) sont liées au choix des critères permettant de discriminer les points skyline et d'en sélectionner les meilleurs suivant le point de vue de l'utilisateur. La quatrième étape comprend l'exécution de l'approche requise pour le raffinement du skyline sélectionné. Enfin, la dernière étape se focalise sur l'évaluation des points skyline retournés et leur validation. Contrairement aux étapes 1, 2, 3 et 5, la quatrième étape ne requiert aucune intervention de l'utilisateur.

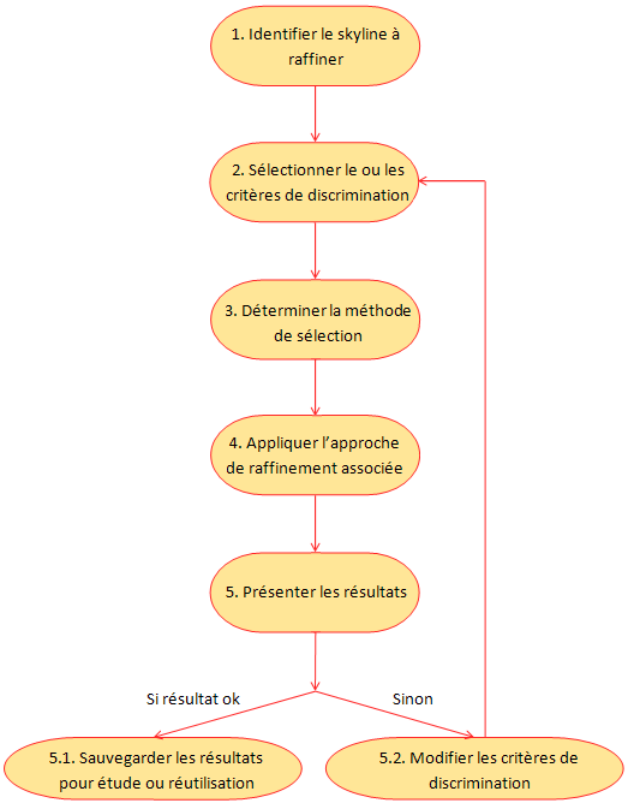



FIGURE 6.3 – Méthodologie de raffinement de skyline.

## 6.5 Analyse de la complexité

Dans cette section, nous étudions la complexité des trois approches de raffinement présentées respectivement dans les sous-sections 6.3.1, 6.3.2 et 6.3.3.

– **La complexité de l'algorithme RDD**

En examinant l'algorithme 6.1 de l'approche RDD, nous constatons que sa complexité temporelle dépend principalement des trois fonctions de calcul suivantes :

La première fonction, notée $F_{RDD-1}$, consiste à calculer les degrés de f-dominance de chaque deux points skyline comparés. La deuxième fonction $F_{RDD-2}$ évalue, quant à elle, les degrés globaux de la f-dominance de chaque point skyline en utilisant par exemple l'opérateur « min ». Enfin, la troisième, notée $F_{RDD-3}$, classe les points skyline des plus intéressants aux moins intéressants selon le critère « nombre des dimensions dominantes ».

La fonction $F_{RDD-1}$ nécessite un temps pour comparer chaque couple de points skyline sur chaque dimension, et calculer le degré avec lequel un point skyline f-domine un autre point du même skyline. Ainsi, pour un skyline composé de « $n$ » points sur « $d$ » dimensions, la complexité de la fonction $F_{RDD-1}$ est de l'ordre $O((n-1).(n/2).d)$. Comme en pratique, la valeur de $d$ est généralement négligeable comparée au nombre de points skyline « $n$ » (i.e., $d \ll n$), la complexité de $F_{RDD-1}$ est donc égale à $O((n-1).(n/2)) \simeq O(n^2)$.

Concernant la fonction $F_{RDD-2}$, le temps nécessaire pour calculer le degré global avec lequel chaque point skyline f-domine tous les autres points skyline, est $O(n.(n-1))) \simeq O(n^2)$. Quant à la fonction $F_{RDD-3}$, sa complexité temporelle résultante de l'ordonnancement des « $n$ » points skyline des plus satisfaisants aux moins satisfaisants suivant leurs degrés globaux de f-dominance, est de l'ordre $O((n-1).(n/2))$. En conséquence, l'approche RDD permettant de raffiner un skyline de taille assez importante selon le nombre des dimensions dominantes, se caractérise par une complexité ***quadratique***.

– **La complexité de l'algorithme MRM**

Dans le cas de l'algorithme 6.2 de l'approche MRM, permettant de raffiner un skyline en considérant à la fois le nombre des dimensions dominantes des points skyline et leurs valeurs sur chaque dimension, son exécution nécessite l'évaluation des cinq fonctions suivantes :

La première fonction, notée $F_{MRM-1}$, calcule de combien la valeur d'un point skyline sur une dimension donnée, est strictement meilleure que celle d'un autre point skyline sur la même dimension (i.e., la fonction $f$ dans l'algorithme 6.2). La deuxième fonction, notée $F_{MRM-2}$, consiste en la fonction $\mu_{mlr}$ de l'algorithme 6.2, permettant d'évaluer le degré avec lequel un point skyline est strictement meilleur qu'un autre point skyline sur chaque dimension.

Quant à la troisième fonction $F_{MRM-3}$, elle considère à la fois le nombre des dimensions dominantes des points skyline et leurs valeurs sur chaque dimension, pour calculer le degré avec lequel un point skyline f-domine un autre point du même skyline. Dans la quatrième fonction, notée $F_{MRM-4}$, une fonction d'agrégation est appliquée pour déterminer dans quelle mesure un point skyline est strictement meilleur que les autres points skyline sur presque toutes les dimensions (i.e., la fonction *Agregation* présentée dans l'algorithme 6.2). Enfin, la cinquième fonction, notée $F_{MRM-5}$, applique une méthode de sélection permettant d'ordonner les points skyline des plus intéressants aux moins intéressants et d'en sélectionner les meilleurs vis-à-vis de l'utilisateur.

Il est facile de vérifier que les complexités temporelles de ces quatre fonctions, pour un skyline de « $n$ » points sur « $d$ » dimensions, sont de l'ordre de $O(n^2)$. Quant à la fonction $F_{MRM-5}$, sa complexité est égale à $O((n-1).(n/2)) \simeq O(n^2)$. En conclusion, l'algorithme 6.2 de l'approche MRM pour raffiner un skyline d'une taille assez élevée est aussi d'une complexité ***quadratique***.

– **La complexité de l'algorithme RID**

Comme l'approche RID n'est qu'une application de la méthode MRM pour raffiner un skyline en considérant un sous-ensemble de dimensions représentant les plus importantes vis-à-dis de l'utilisateur, il est facile de vérifier que la complexité de son algorithme est également ***quadratique***.

En conclusion, les approches proposées dans ce chapitre pour raffiner un skyline de taille importante, ont une complexité ***quadratique*** (i.e., *polynomiale*). Ce qui signifie que le coût de leur application reste acceptable.

## 6.6 Conclusion

Nous nous sommes intéressés dans ce chapitre au problème de raffinement de skyline lorsque ce dernier est d'une taille assez importante. Nous avons présenté trois approches de raffinement fondées sur une nouvelle relation de dominance plus restrictive que celle de Pareto. Cette relation, appelée *f-dominance*, permet de discriminer les points skyline et les rendre comparables en se basant sur le quantificateur linguistique « *presque tous* ».

Chaque approche de raffinement proposée utilise des critères de discrimination permettant ainsi de sélectionner les points skyline qui répondent le mieux à la requête de l'utilisateur sur presque toutes les dimensions. La première approche RDD considère le nombre des dimensions dominantes entre chaque deux points skyline comparés. La deuxième, appelée MRM, renforce la première approche en ajoutant un autre critère permettant d'améliorer la discrimination des points skyline. Pour ce faire, son processus de raffinement prend en considération à la fois le nombre des dimensions dominantes et les valeurs des points skyline sur chaque dimension. Quant à la troisième approche RID, elle considère, en plus des critères de discrimination utilisés dans la méthode MRM, l'ensemble des dimensions les plus importantes vis-à-vis de l'utilisateur.

Le skyline résultant de chaque approche est d'une nature graduelle, i.e., chaque point skyline est associé à un degré exprimant dans quelle mesure ce dernier *est strictement meilleur que tous les autres points du même skyline sur presque toutes les dimensions*. De cette façon, les points skyline peuvent être ordonnés pour que l'utilisateur choisisse soit les top-k points skyline, soit ceux qui ont un degré supérieur ou égal à un certain seuil $\sigma$ (défini par l'utilisateur).

Nous avons également introduit dans ce chapitre, une méthodologie de raffinement de skyline à partir des trois approches : RDD, MRM et RID. Enfin, une analyse de la complexité temporelle de ces dernières a été réalisée. Elle est de nature *quadratique* et donc leur utilisation ne résulte pas en un coût excessif.

Il est important de préciser que les approches de raffinement proposées s'adaptent facilement à d'autres types de quantificateurs linguistiques tels que le quantificateur « *au moins* $k$ », où $k$ est supérieur ou égal à 2 dimensions.

# Conclusion générale

# Conclusion générale

Pour conclure ce manuscrit, nous présentons dans ce qui suit un résumé des travaux que nous avons réalisés au cours de cette thèse et qui portent, d'une manière générale, sur l'amélioration du processus d'appariement d'objets complexes et donc de la pertinence des résultats retournés. Les perspectives de recherche qui pourraient être envisagées suite à ces travaux sont également discutées.

## Synthèse

Plus précisément, ces travaux contribuent à l'accès personnalisé et coopératif d'objets complexes modélisés sous forme de graphes. Ces objets, de plus en plus abondants, sont produits par de nombreuses applications. L'ensemble de ces travaux se fondent sur des outils mathématiques de la théorie des ensembles flous et s'intègrent dans le cadre du projet ANR-AOC [1] (Appariement d'Objets Complexes).

Dans un premier temps, nous nous sommes intéressés à la problématique de recherche de services Web modélisés par des graphes (appelés aussi graphes de modèles de processus). Nous avons proposé, en collaboration avec l'équipe PRiSM de l'Université de Versailles Saint-Quentin-en-Yvelines, une amélioration d'un *système de sélection de services Web* qui permet de combiner les contraintes fonctionnelles (i.e., sur l'aspect structurel des graphes) et non-fonctionnelles (i.e., sur les aspects de qualité de services). La prise en compte de ces deux aspects permet de réduire le nombre de services Web candidats offrant des fonctionnalités et des comportements similaires et d'ordonner les réponses des plus pertinentes aux moins pertinentes par rapport aux préférences de l'utilisateur. De plus, la méthodologie appliquée dans le système de sélection de services Web permet aussi de réduire l'espace de recherche du processus d'appariement structurel lorsqu'une mise en correspondance entre deux activités de deux graphes ne permet pas la vérification d'une contrainte non-fonctionnelle. L'analyse de la complexité de l'approche proposée a montré qu'elle est de nature polynomiale et donc acceptable comparée à la complexité de l'algorithme d'appariement structurel, d'une manière générale. Enfin, les expérimentations réalisées pour valider et évaluer l'approche proposée ont montré que : (i) le temps supplémentaire nécessaire pour évaluer les préférences utilisateur est négligeable ; (ii) l'évaluation des préférences n'impacte pas le temps d'exécution du processus d'appariement structurel et (iii) les métriques proposées contribuent à l'amélioration de la pertinence des résultats retournés.

1. http ://aoc.irit.fr/.

Dans un second temps, nous avons proposé l'approche « *skyline de graphes par similarité* », qui est fondée sur la relation de dominance au sens de Pareto utilisant un vecteur de distances entre graphes. Les avantages majeurs de cette approche sont : (i) la possibilité d'affranchir l'utilisateur du choix souvent difficile d'une mesure de calcul de distance complexe entre graphes parmi celles proposées dans la littérature ; (ii) la préservation des propriétés et des caractéristiques de chacune des mesures du calcul de distances utilisées et enfin (iii) la sélection des graphes maximalement similaires au graphe requête de l'utilisateur. Cette approche peut donc être naturellement intégrée dans la phase de vérification de tout système d'interrogation de bases de données de graphes. Pour valider et évaluer notre approche, un prototype, appelé « *GraphSim* », a été développé permettant la sélection des graphes les plus similaires au graphe requête de l'utilisateur en utilisant un vecteur de mesures de distance. Nous avons également présenté une méthode de raffinement permettant de réduire la taille du skyline de graphes lorsque ce dernier est d'une taille importante. Cette méthode est basée sur le critère de diversité pour sélectionner à partir de skyline de graphes par similarité un sous-ensemble de taille « k » le plus divers possible.

Nous nous sommes intéressés par la suite au problème de raffinement du skyline, en général. Trois approches ont été proposées et qui sont basées sur une nouvelle variante de dominance qui est plus sélective que celle de Pareto. Elles permettent de sélectionner les points skyline qui répondent le mieux à la requête de l'utilisateur sur un maximum de dimensions. Le concept clé de la nouvelle relation de dominance introduite est les quantificateurs linguistiques. Nous avons décrit une méthodologie de raffinement qui peut être intégrée dans un système d'interrogation de bases de données multidimensionnelles en général. Enfin, l'analyse de la complexité temporelle de nos approches de raffinement de skyline a été examinée. La complexité globale de chaque approche reste polynomiale.

## Perspectives

Les perspectives suscitées par ces travaux sont multiples. Une partie d'entre elles concernent les approches liées aux graphes modélisant les modèles de processus et la seconde vise à étendre les travaux proposés relatifs au paradigme skyline.

- **Enrichissement des expérimentations.** Un des aspects qui nous semble important à améliorer dans les expérimentations menées est d'accroître le nombre d'experts consultés. Ceci permet, d'une part, de mieux tester la qualité des mesures suggérées et, d'autre part, de construire un benchmark plus complet et reflétant plus la réalité.
- **Définition de préférences structurelles.** À travers le prototype « *MatchMaker* » de PRiSM, nous avons montré la faisabilité et la pertinence de notre approche de sélection de services Web, en ajoutant des préférences utilisateur sur les aspects de qualité de service (QdS). Un autre moyen permettant d'améliorer la sélectivité des systèmes de recherche de services Web en particulier, est de considérer des contraintes exprimées au niveau structurel des graphes. En d'autres termes, l'intégration de contraintes sur la composition structurelle des graphes similaires au graphe requête de l'utilisateur permet d'améliorer non seulement l'efficacité du

processus d'appariement en éliminant les graphes qui ne vérifient pas ces contraintes mais aussi la pertinence des résultats en renvoyant les réponses servant le mieux les attentes de l'utilisateur.

- **Proposition de nouvelles mesures d'agrégation.** Dans l'approche de sélection de services Web, les mesures proposées pour évaluer les degrés de satisfaction des préférences, de similarité structurelle et de similarité globale entre les modèles de processus sont basées sur la théorie des ensembles flous et permettent d'améliorer la pertinence des résultats et d'offrir une sémantique interprétable à ces degrés. Toutefois, d'autres types de mesures d'agrégation peuvent être appliqués, tels que l'intégrale de Sugeno [125] qui a la particularité de prendre en compte non seulement la satisfaction des préférences sur chaque critère et leur niveau d'importance mais aussi l'importance des coalitions entre les critères. L'intégrale de Sugeno est un opérateur d'agrégation ordinal qui permet une évaluation qualitative des requêtes composées de préférences conjonctives.
- **Amélioration de l'approche de raffinement de skyline de graphes par similarité.** L'approche de raffinement proposée dans le chapitre 5 permet de réduire la taille du skyline en sélectionnant le sous-ensemble de taille « k » le plus divers possible. Pour cela, elle doit calculer la distance entre chaque couple de graphes de chaque sous-ensemble de taille « k » du skyline par similarité $S$, ce qui demande un temps d'exécution assez important au processus d'appariement entre graphes. Ainsi, une heuristique est nécessaire pour définir une méthode permettant d'améliorer le temps d'exécution de l'approche de raffinement proposée.
- **Mise en œuvre des approches de raffinement de skyline RDD, MRM et RID.** Nous envisageons dans les travaux futurs d'implémenter un prototype permettant d'évaluer et de valider les approches de raffinement de skyline RDD, MRM et RID que nous avons proposées dans le chapitre 6. Ceci nous servira de base pour évaluer la performance et la pertinence de ces méthodes. Une seconde extension de ces approches est d'utiliser d'autres types de quantificateurs linguistiques comme « au moins k ».
- **Relaxation de skyline.** Des méthodes de relaxation de skyline sont nécessaires lorsque ce dernier est d'une taille insuffisante du point de vue de l'utilisateur. D'une manière similaire aux méthodes RDD, MRM et RID, il est également possible d'appliquer la notion de quantificateurs linguistiques tels que « presque tous » et « au moins k », pour sélectionner les points qui se rapprochent le plus aux points skyline.

# Bibliographie


[1] K. Abbaci, A. Hadjali, L. Liétard et D. Rocacher : A similarity skyline approach for handling graph queries - a preliminary report. *In GDM*, 2011.

[2] K. Abbaci, A. Hadjali, L. Liétard et D. Rocacher : Un opérateur skyline flexible pour l'interrogation de bases de données : Premières explorations. *In Proceedings of LFA'2012 (Logique Floue et ses Applications)*, Compiègne, France, November 15th - 16th 2012.

[3] K. Abbaci, A. Hadjali, L. Liétard et D. Rocacher : A linguistic quantifier-based approach for skyline refinement. *In North American Fuzzy Information Processing Society (IFSA'13)*, 2013.

[4] K. Abbaci, F. Lemos, A. Hadjali, D. Grigori, L. Liétard, D. Rocacher et M. Bouzeghoub : Selecting and ranking business processes with preferences : An approach based on fuzzy sets. *In CoopIS*, pages 38 – 55, 2011.

[5] Katia Abbaci, Fernando Lemos, Allel Hadjali, Daniel Grigori, Ludovic Liétard, Daniel Rocacher et Mokrane Bouzeghoub : An approach based on fuzzy sets to selecting and ranking business processes. *In Proceedings of the IEEE Conference on Commerce and Enterprise Computing (CEC'11)*, pages 213 – 218, Luxembourg, September 5th - 7th 2011.

[6] Katia Abbaci, Fernando Lemos, Allel Hadjali, Daniel Grigori, Ludovic Liétard, Daniel Rocacher et Mokrane Bouzeghoub : A cooperative answering approach to fuzzy preferences queries in service discovery. *In Proceedings of the 9th International Conference on Flexible Query Answering Systems (FQAS'11)*, pages 318 – 329, Ghent, Belgium, October 26th - 28th 2011.

[7] Katia Abbaci, Fernando Lemos, Allel Hadjali, Daniel Grigori, Ludovic Liétard, Daniel Rocacher et Mokrane Bouzeghoub : A fuzzy set-based approach to handle preferences in service retrieval. *In BDA'11 (27èmes journées Bases de Données Avancées)*, Rabat, Maroc, October 24th - 27th 2011.

[8] Katia Abbaci, Fernando Lemos, Allel Hadjali, Daniel Grigori, Ludovic Liétard, Daniel Rocacher et Mokrane Bouzeghoub : Selecting and ranking business processes with preferences : An approach based on fuzzy sets. *In Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS'11)*, pages 38 – 55, Crete, Greece, October 19th - 21st 2011.

[9] Katia Abbaci, Fernando Lemos, Allel Hadjali, Daniel Grigori, Ludovic Liétard, Daniel Rocacher et Mokrane Bouzeghoub : A bipolar approach to the handling of user preferences in business processes retrieval. *In Proceedings of the 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'12)*, Catania, Italy, July 9th - 13th 2012.

[10] Eyhab Al-Masri et Qusay H. Mahmoud : Qos-based discovery and ranking of web services. *In Proceedings of 16th International Conference on Computer Communications and Networks (ICCCN 2007)*, pages 529 – 534, 2007.

[11] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic et S. Weerawarana : *BPEL4WS, Business Process Execution Language for Web Services Version 1.1.* IBM, 2003.

[12] Egon Balas et Chang Sung Yu : Finding a maximum clique in an arbitrary graph. *Society for Industrial and Applied MathematicsJournal on Computing*, 15(4):1054–1068, 1986.

[13] S. Benferhat, D. Dubois, S. Kaci, et H. Prade : Bipolar possibility theory in preference modeling : Representation, fusion and optimal solution. *IJIF*, pages 135 – 150, 2006.

[14] Stephan Borzsonyi, Donald Kossmann et Konrad Stocker : The skyline operator. *In Proceedings of the 17th International Conference on Data Engineering*, pages 421 – 430, 2001.

[15] P. Bosc, L. Liétard, O. Pivert et D. Rocacher : *Gradualité et imprécision dans les bases de données : Ensembles flous, requêtes flexibles et interrogation de données mal connues.* Numéro 320. Ellipses marketing édition, 2004.

[16] P. Bosc et O. Pivert : Sqlf : a relational database language for fuzzy querying. *IEEE Trans. on Fuzzy Systems*, 3(1):1 – 17, 1995.

[17] Horst Bunke : On a relation between graph edit distance and maximum common subgraph. *Pattern Recogn Letters*, 18 (9):689–697, August 1997.

[18] Horst Bunke : Error correcting graph matching : On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917 – 922, 1999.

[19] Horst Bunke, Pasquale Foggia, C. Guidobaldi, Carlo Sansone et Mario Vento : A comparison of algorithms for maximum common subgraph on randomly connected graphs. *In International Workshops Structural, Syntactic, and Statistical Pattern Recognition*, pages 123 – 132, Windsor, Ontario, Canada, August 6th - 9th 2002.

[20] Horst Bunke et Kim Shearer : A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19 (3-4):255–259, March 1998.

[21] Host Bunke et G Allerman : Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245 – 253, May 1983.

[22] Host Bunke, Xiaoyi Jiang et Abraham Kandel : On the minimum common supergraph of two graphs. *Computing*, 65(1):13 – 25, 2000.

[23] D. Cai, Z. Shao, X. He, X. Yan et J. Han : Community mining from multirelational networks. *In Proc. of PPKDD*, pages 445–452, 2005.

[24] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung et Z. Zhang : Finding k-dominant skylines in high dimensional space. *In SIGMOD*, 2006.

[25] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung et Z. Zhang : On high dimensional skylines. *In EDBT*, 2006.

[26] Chen Chen, Xifeng Yan, Philip S. Yu, Jiawei Han, Dong-Qing Zhang et Xiaohui Gu : Towards graph containment search and indexing. *In Proceeding of the 33rd International Conference on Very Large Data Bases*, pages 926–937, Vienna, Austria, sept 23-27 2007.

[27] T.-Y. Chen et J.-C. Wang : Identification of $\lambda$-fuzzy measures using sampling design and genetic algorithms. *Fuzzy Sets Systems*, 123(3):321 – 341, 2001.

[28] James Cheng, Yiping Ke et Wilfred NG : Efficient query processing on graph databases. *ACM on Transactions on Database Systems (TODS)*, 34, 2009.

[29] James Cheng, Yiping Ke, Wilfred Ng et An Lu : Fg-index : Towards verification-free query processing on graph databases. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 857–872, 2007.

[30] Jan Chomicki : Preference formulas in relational queries. *ACM Transactions on Database Systems*, 27:153–187, 2003.

[31] G. Choquet : Theory of capacities. *Annales de l'Institut Fourier*, 5:131 – 295, 1953.

[32] E. F. Combarro et P. Miranda : Identification of fuzzy measures from sample data with genetic algorithms. *Computers & Operations Research*, 33 (10):3046 – 3066, 2006.

[33] D. Conte, P. Foggia, C. Sansone et M. Vento : Thirty years of graph matching in pattern recognition. *Inter. J. of Pattern Recogn. and Art. Intell.*, 18 (3):265–298, 2004.

[34] Donatello Conte, Pasquale Foggia et Mario Vento : Challenging complexity of maximum common subgraph detection algorithms : A performance analysis of three algorithms on a wide database of graphs. *Journal of Graph Algorithms and Applications*, 11(1):99 – 143, 2007.

[35] Donatello Conte, C. Guidobaldi et Carlo Sansone : A comparison of three maximum common subgraph algorithms on a large database of labeled graphs. *In International Workshop on Graph-based Representation in Pattern Recognition*, pages 130 – 141, York, UK, June 30 - July 2 2003.

[36] I. Şora, G. Lazăr et S. Lung : Mapping a fuzzy logic approach for qos-aware service selection on current web service standards. *In ICCC-CONTI*, pages 553 – 558, 2010.

[37] B. Cuissart : *Plus grande structure commune à deux graphes : méthode de calcul et intérêt dans un contexte SAR.* Thèse de doctorat, Université de Caen/Basse-Normandie, December 2004.

[38] B. Cuissart : Plus grand sous-graphe induit et connexe commun à deux graphes : méthode de calcul et application en traitement de l'information chimique,. Rapport technique, Groupe de

Recherche en Informatique, Image, Automatique et Instrumentation de Caen, France (CNRS - UMR 6072), Janvier 2007.

[39] Guy de Tré, Slawomir Zadrożny et Antoon Bronselaer : Handling bipolarity in elementary queries to possibilistic databases. *In IEEE Transaction on Fuzzy Systems*, volume 18(3), pages 599 – 612, 2010.

[40] Guy de Tré, Slawomir Zadrożny, Tom Matthé, Janusz Kacprzyk et Antoon Bronselaer : Dealing with positive and negative query criteria in fuzzy database quering bipolar satisfaction degrees. *In FQAS*, pages 593 – 604, 2009.

[41] R. Dijkman, M. Dumas et L. García-Bañuelos : Graph matching algorithms for business process model similarity search. *In Proc. of BPM*, pages 48 – 63, 2009.

[42] B. Dongen, R. Dijkman et J. Mendling : Measuring similarity between business process models. *In Proc. of CAISE*, pages 450 – 464, 2008.

[43] Grégoire Dooms, Yves Deville et Pierre Dupont : Constrained metabolic network analysis : discovering pathways using cp(graph). *In Workshop on Constraint Based Methods for Bioinformatics*, 2005.

[44] D. Dubois, J.-L. Marichal, H. Prade, M. Roubens et R. Sabbadin : The use of the discrete sugeno integral in decision making : a survey. *Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(5):539 – 561, 2001.

[45] D. Dubois et H. Prade : Using fuzzy sets in flexible querying : Why and how ? *In FQAS*, pages 89 – 103, 1996.

[46] D. Dubois et H. Prade : Bipolarité dans un processus d'interrogation flexible. *In LFA*, 2002.

[47] D. Dubois et H. Prade : Bipolarity in flexible querying. *In FQAS*, volume 2522, pages 174 – 182, 2002.

[48] D. Dubois et H. Prade : *Handling Bipolar Queries in Fuzzy Information Processing*, chapitre Fuzzy Information Processing in Databases, pages 97 – 114. 2008.

[49] Marlon Dumas, Luciano García-Bañuelos, Artem Polyvyanyy, Yong Yang et Liang Zhang : Aggregate quality of service computation for composite services. *In* Paul P. Maglio, Mathias Weske, Jian Yang et Marcelo Fantinato, éditeurs : *ICSOC*, volume 6470 de *Lecture Notes in Computer Science*, pages 213–227, 2010.

[50] M. Endres et W. Kiessling : Skyline snippets. *In FQAS*, pages 246 – 257, 2011.

[51] M. Eshera et King-Sun Fu : A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(3):398 – 408, 1984.

[52] Stefan Fankhauser, Kaspar Riesen et Horst Bunke : Speeding up graph edit distance computation through fast bipartite matching. *In* Xiaoyi Jiang, Miquel Ferrer et Andrea Torsello, éditeurs : *Graph-Based Representations in Pattern Recognition*, volume 6658 de *Lecture Notes in Computer Science*, pages 102–111, Münster, Germany, May 18 - 20 2011. Springer.

[53] M. Farina et P. Amato : Fuzzy optimality and evolutionary multiobjective optimization. *In EMO*, pages 58 – 72. Springer, 2003.

[54] Mirtha-Lina Fernández et Gabriel Valiente : A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22 (7-6):753–758, May 2001.

[55] Xinbo Gao, Bing Xiao, Dacheng Tao et Xuelong Li : A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113 – 129, 2010.

[56] M. Garey, D. Johnson et Javier Bezos : Computers and intractability : A guide to the theory of np-completeness. Rapport technique, W. H. Freeman and Company, 1979.

[57] Ahmed Gater : *Process matching and discover*. Thèse de doctorat, University of Versailles Saint-Quentin-en-Yvelines, France, 2012.

[58] Ahmed Gater, Daniela Grigori et Mokrane Bouzeghoub : Matching and similarity evaluation of owl-s process models. *In 25èmes Journées de Bases de Données Avancées*, Namur, Belgique, 2009.

[59] Ahmed Gater, Daniela Grigori et Mokrane Bouzeghoub : A graph-based approach for semantic process model discovery. *In Graph Data Management*, pages 438 – 462. 2011.

[60] Rosalba Giugno et Dennis Shasha : Graphgrep : A fast and universal method for querying graphs. *In Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 112–115, 2002.

[61] I. Glöckner : *Fuzzy Quantifiers in Natural Language : Semantics and Computational Models*. Der Andere Verlag, Osnabrück, Germany, juillet 2004.

[62] M. Goncalves et L. J. Tineo : Fuzzy dominance skyline queries. *In DEXA*, volume 4653, 2007.

[63] M. Grabisch : The application of fuzzy integrals in multicriteria decision making. *European J. of Operational Research*, 89:445 – 456, 1996.

[64] M. Grabisch et P. Perny : Agrégation multicritère. *In* B. Bouchon et C. Marsala, éditeurs : *Utilisations de la logique floue*. Hermès, 1999.

[65] Michel Grabisch : Fuzzy integral in multicriteria decision making. *Fuzzy Sets & Systems*, 69(3):279 – 298, 1995.

[66] Michel Grabisch, Sergei A. Orlovski et Ronald R. Yager : *Fuzzy Aggregation of Numerical Preferences*, chapitre 1. 1998.

[67] Daniela Grigori, Juan Carlos Corrales, Mokrane Bouzeghoub et Ahmed Gater : Ranking bpel processes for service discovery. *IEEE Transactions on Services Computing*, 3:178–192, 2010.

[68] A. Hadjali, O. Pivert et H. Prade : On different types of fuzzy skylines. *In ISMIS*, 2011.

[69] A. Hadjali, O. Pivert et H. Prade : Possibilistic contextual skylines with incomplete preferences. *In Proc. of SoCPaR, Cergy Pontoise, Paris, France*, December 07-10, 2010.

[70] Allel Hadjali, Souhila Kaci et Henri Prade : Database preferences queries - a possibilistic logic approach with symbolic priorities. *In Proceedings of the 5th International Symposium on Foundations of Information and Knowledge Systems*, pages 291 – 310, 2008.

[71] H. He et A. K. Singh : Closure-tree : An index structure for graph queries. *In Proceedings of the 22nd International Conference on Data Engineering*, page 38, 2006.

[72] Almut Herzog, Nahid Shahmehri et Claudiu Duma. : An ontology of information security. *International Journal of Information Security and Privacy (IJISP)*, 1(4):1 – 23, 2007.

[73] Eyke Hüllermeier, Ilya Vladimirskiy, Belén Prados Suárez et Eva Stauch : Supporting case-based retrieval by similarity skylines : Basic concepts and extensions. *In Proceedings of European Conference on Case-Based Reasoning*, pages 240 – 254, 2008.

[74] J. E. Hopcroft et J. K. Wong : Linear time algorithm for isomorphism of planar graphs (preliminary report). *In STOC'74 : Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 172 – 184, New York, USA, 1974.

[75] H. Hu, Y. Hang, J. Han et X.J. Zhou : Mining coherent dense subgraphs across massive biological network for fonctional discovery. *Bioinformatics*, 1(1):1–9, 2005.

[76] E. Hüllermeier, I. Vladimirskiy, B. Prados SuŽarez et E. Stauch : Supporting case-based retrieval by similarity skylines : Basic concepts and extensions. *In ECCBR*, volume 5239, pages 240 – 254, 2008.

[77] Haoliang Jiang, Hainun Wang, Philip S. Yu et S. Zhou : Gstring : A novel approach for efficient search in graph databases. *In Proceedings of the International Conference on Data Engineering*, pages 566–575, 2007.

[78] Derek Justice et Alfred Hero : A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200 – 1214, 2006.

[79] Mohamed E. Khalefa, Mohamed F. Mokbel et Justin J. Levandoski : Skyline query processing for incomplete data. *In Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 556 – 565, 2008.

[80] Werner Kiessling : Foundations of preferences in database systems. *In VLDB*, pages 311 – 322, 2002.

[81] Werner Kiessling et Gerhard Köstler : Preference sql - design, implementation, experiences. *In VLDB*, pages 990 – 1001, 2002.

[82] S. Klinger et J. Austin : Chemical similarity searching using a neural graph matcher. *In Proc. of ESANN*, pages 479–484, 2005.

[83] G. J. Klir, Z. Wang et D. Harmanec : Constructing fuzzy measures in expert systems. *Fuzzy Sets Systems*, 92 (2):251 – 264, 1997.

[84] Matthias Klusch, Benedikt Fries et Katia Sycara : Automated semantic web service discovery with owls-mx. *In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 915–922, 2006.

[85] Evgeny B. Krissinel et Kim Henrick : Common subgraph isomorphism detection by backtracking search. *Software : Practice and Experience*, 34(6):591 – 607, 2004.

[86] K. Kritikos et D. Plexousakis : Semantic qos metric matching. *In Proc. of ECOWS*, pages 265 – 274, 2006.

[87] S. Kukkonen et J. Lampinen : Ranking-dominance and many-objective optimization. *In IEEE Congress on Evolutionary Computation*, pages 3983–3990, 2007.

[88] Michihiro Kuramochi et George Karypis : Frequent subgraph discovery. *In 1st IEEE International Conference on Data Mining,*, pages 313 – 320, 2001.

[89] Cyril Laitang et Karen Pinel-Sauvagnat : Utilisation de la théorie des graphes et de la distance d'édition pour la recherche d'information sur documents xml (regular paper). *In Proceedings of CORIA'11 (Conférence francophone en Recherche d'Information et Applications)*, pages 349 – 364, Avignon, France, march 16th - 18th 2011.

[90] F. Lemos, A. Gater, D. Grigori et M. Bouzeghoub : Adding preferences to semantic process model matchmaking. *In Proc. of GAOC*, 2011.

[91] Fernando Lemos, Katia Abbaci, Allel Hadjali, Daniel Grigori, Mokrane Bouzeghoub, Ludovic Liétard et Daniel Rocacher : Intégration de préférences dans la découverte et la sélection des processus métiers : une approche fondée sur les ensembles flous. *ISI'11 (Revue d'Ingénierie des Systèmes d'Information)*, 2011.

[92] Fernando Cordeiro De Lemos : *Infrastructure et algorithmes pour l'analyse de la qualité d'information et pour la découverte de processus.* Thèse de doctorat, Université de Versailles Saint-Quentin-en-Yvelines, 18 Février 2013.

[93] Vladimir I. Levenshtein : Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707 – 710, 1966.

[94] Xian Tong Li et Jian Zhon Li : An efficient graph query algorithm based on important vertices and decision features. *Internationanl Journal of Information Technology*, 1(5), 2009.

[95] L. Liétard : *Contribution à l'interrogation flexible de bases de données : Étude des propositions quantifiées floues.* Thèse de doctorat, Université de Rennes 1, 1995.

[96] L. Liétard : A functional interpretation of linguistic summaries of data. *In Information Sciences*, volume 188, pages 1–16, 2012.

[97] L. Liétard, D. Rocacher et N. Tamani : A relational algebra for generalized fuzzy bipolar conditions. *In* O. Pivert et S. Zadrozny eds, éditeurs : *Flexible Approaches in Data, Information and Knowledge Management*, volume 497 de *Studies in Computational Intelligence*, pages 45 – 69. Springer, 2013.

[98] L. Liétard, N. Tamani et D. Rocacher : Fuzzy bipolar conditions of type "or else". *In Fuzzy Systems (FUZZ), 2011 IEEE International Conference*, 2011.

[99] Xuemin Lin, Yidong Yuan, Qing Zhang et Ying Zhang : Selecting stars : The k most representative skyline operator. *In Proceedings of the 23rd International Conference on Data Engineering*, pages 86 – 95, 2007.

[100] J.-L. Marichal : On sugeno integral as an aggregation function. *Fuzzy Sets & Systems*, 114:347 – 365, 2000.

[101] D. Martin, M. Burstein, J. Hobbs, O. Lassila, Dr. McDermott, Sh. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. R. Payne, E. Sirin, N. Srinivasan et K. Sycara : Owl-s : Semantic markup for web services. 2004.

[102] James J. McGregor : Backtrack search algorithms and the maximal common subgraph problem. *Software : Practice and Experience*, 12(1):23 – 34, 1982.

[103] David McSherry : Diversity-conscious retrieval. *In Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 219–233. Springer-Verlag, 2002.

[104] B. T. Messmer et H. Bunke : A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:493 – 504, 1998.

[105] S. Ben Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny et Y. Berbers : Easy : Efficient semantic service discovery in pervasive computing environments with qos and context support. *Journal of Systems and Software*, 81(5):785 – 808, 2008.

[106] T. Murofushi et M. Sugeno : An interpretation of fuzzy measure and the choquet integral as an integral with respect to fuzzy measure. *Fuzzy Sets and Systems*, 29:201 – 227, 1989.

[107] Michel Neuhaus et Horst Bunke : A probabilistic approach to learning costs for graph edit distance. *In Proceedings of the 17th International Conference on Pattern Recognition*, pages 389 – 393, 2004.

[108] Michel Neuhaus et Horst Bunke : Automatic learning of cost functions for graph edit distance. *Information Sciences*, 117(1):239 – 247, 2007.

[109] Michel Neuhaus, Kaspar Riesen et Horst Bunke : Fast suboptimal algorithms for the computation of graph edit distance. *In STRUCTURAL, SYNTACTIC, AND STATISTICAL PATTERN RECOGNITION. LNCS*, pages 163 – 172. Springer, 2006.

[110] D. Papadias, Y. Tao, G. Fu et B. Seeger : An optimal and progressive algorithm for skyline queries. *In SIGMOD*, 2003.

[111] Jian Pei, Bin Jiang, Xuemin Lin et Yidong Yuan : Probabilistic skylines on uncertain data. *In Proceedings of the 33rd international conference on Very large data bases*, pages 15 – 26, 2007.

[112] Marcello Pelillo, Kaleem Siddiqi et Steven W. Zucker : Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:1105 – 1120, 1998.

[113] Euripides G. M. Petrakis et Christos Faloutsos : Similarity searching in medical image databases. *IEEE Transactions on Knowledge and Data Engineering*, 9 (3):435 –447, May 1997.

[114] John W. Raymond et Peter Willett : Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16(7): 521 – 533, 2002.

[115] Kaspar Riesen, Michel Neuhaus et Horst Bunke : Bipartite graph matching for computing the edit distance of graphs. *Graph-Based Representations in Pattern Recognition*, pages 1 – 12, 2007.

[116] Alberto Sanfeliu et King-Sun Fu : A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 13(3):353 – 362, May 1983.

[117] Stanley M. Selkow : The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184 – 186, December 1977.

[118] Haichuan Shang, Ying Zhang, Xuemin Lin et Jeffrey Xu Yu : Taming verification hardness : an efficient algorithm for testing subgraph isomorphism. *PVLDB*, 1(1):364 – 375, 2008.

[119] Haichuan Shang, Ke Zhu, Xuemin Lin, Ying Zhang et Ryutaro Ichise : Similarity search on supergraph containment. *In 26th IEEE International Conference on Data Engineering*, pages 637–648, march 1-6 2010.

[120] Dennis Shasha, Jason T. L. Wang et Rosalba Giugno. : Algorithmics and applications of tree and graph searching. *In Proceeding 21th ACM Symp. principles of Database Systems*, pages 39–52, Jun 2002.

[121] K. Shearer, H. Bunke, S. Venkatesh et D. Kieronska : Efficient graph matching for video indexing. *In Graph based representations in pattern recognition*, pages 53 – 62, 1997.

[122] Nicolas Sidère : *Contribution aux méthodes de reconnaissance structurelle de formes : approche à base de projection de graphes.* Thèse de doctorat, Université de François Rabelais de Tours, 2012.

[123] A. Soria-Frisch : Unsupervised construction of fuzzy measures through self-organizing feature maps and its application in color image segmentation. *International Journal of Approximate Reasoning*, 41:23 – v42, 2006.

[124] Sébastien Sorlin : *Mesurer la similarité de graphes.* Thèse de doctorat, Université Claude Bernard Lyon I, 2006.

[125] M. Sugeno : *Theory of fuzzy integrals and its applications.* Thèse de doctorat, Tokyo Institute of Technology, 1974.

[126] M. Sugeno : Fuzzy measures and fuzzy integrals : a survey. *Fuzzy Automata and Decision Processes*, pages 89 – 102, 1977.

[127] H. Tahani et J. M. Keller : Information fusion in computer vision using the fuzzy integral. *IEEE Trans. SMC*, 20 (3):733 – 741, 1990.

[128] Kuo C. Tai : The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26(3):422 – 433, 1979.

[129] E. Takahagi : A fuzzy measure identification method by diamond pairwise comparisons and $\phi$s transformation. *Fuzzy Optimization and Decision Making*, 7:219 – 232, 2008.

[130] N. Tamani, L. Liétard et D. Rocacher : Bipolar sqlf : a flexible querying language for relational databases. *In* H. Christiansen et al. eds, éditeur : *The 9th International Conference on Flexible Query Answering Systems (FQAS'11)*, volume 7022 de *Lecture Notes in Computer Science*, pages 472 – 484. Springer, 2011.

[131] Nouredine Tamani : *Interrogation personnalisée des systèmes d'information dédiés au transport : une approche bipolaire floue*. Thèse de doctorat, Université de Rennes 1, 2012.

[132] Y. Tian, R. C. McEachin, C. Santos, D. J. States et J. M. Patel : Saga : a subgaph matching tool for biological graphs. *Bioinformatic*, 23(2):232–239, 2007.

[133] Y. Tian et J. M. Patel : Tale : A tool for approximate large graph matching. *In Proceeding of the International Conference on Data Engineering, Cancun, Mexico*, pages 963–972, 2008.

[134] Li Xian Tong et Li Jian Zhong : A decomposition based algorithm of graph containment query. *Information Technology Journal 8*, 2:214–219, 2010.

[135] Wen-Hsiang Tsai et King-Sun Fu : Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:757 – 768, 1979.

[136] A. S. Wagholikar et J. Jo : An intelligent system in healthcare using fuzzy measures. *In ISKE*, 2008.

[137] W. D. Wallis, P. Shoubridge, M. Kraetz et D. Ray : Graph distances using graph union. *Pattern Recognition Letters*, 22 (6-7):701–704, May 2001.

[138] Yu Wang et Carsten Maple : A novel efficient algorithm for determining maximum common subgraphs. *In Ninth International Conference on In Information Visualisation*, pages 657 – 663, 2005.

[139] P. Willett, J. Barnard et G Downs : Chemical similarity searching. *Journal of ChemInformatic Computing Sciences*, 38 (6):983–996, 1998.

[140] D. W. Williams, J. Huan et W.i Wang : Graph database indexing using structured graph decomposition. *In Proceedings of the International Conference on Data Engineering*, pages 976–985, 2007.

[141] Z. Wu et M. S. Palmer : Verb semantics and lexical selection. *In Proc. of ACL*, pages 133 –138, 1994.

[142] R. R. Yager : General multiple-objective decision functions and linguistically quantified statements. *International Journal of Man-Machine Studies*, 21:389 – 400, 1984.

[143] R. R. Yager : Fuzzy quotient operators for fuzzy relational databases. *In Fuzzy Engineering Symp. (IFES'91)*, pages 289 – 296, 1991.

[144] Xifeng Yan et Jiawei Han : Closegraph : Mining closed frequent graph paterns. *In Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining*, pages 286 – 295, 2003.

[145] Xifeng Yan, Philip S. Yu et Jiawei Han : Graph indexing : A frequent structurebased approach. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 335–346, 2004.

[146] Xifeng Yan, Philip S. Yu et Jiawei Han : Substructure similarity search in graph databases. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 766–777, 2005.

[147] Man Lung Yiu et Nikos Mamoulis : Efficient processing of top-k dominating queries on multi-dimensional data. *In Proceedings of the 33rd international conference on Very large data bases*, pages 483–494, 2007.

[148] L. A. Zadeh : Fuzzy sets. *Information and Control*, 8:338 – 353, 1965.

[149] L. A. Zadeh : A computational approach to fuzzy quantifiers in natural languages. *International series in modern applied mathematics and computer science*, 9:149 – 183, 1983.

[150] S. Zadrożny et J. kacprzyk : Bipolar queries and queries with preferences (invited paper). *In DEXA'06*, 2006.

[151] S. Zadrożny et J. Kacprzyk : Bipolar queries using various interpretations of logical connectives. *In LNAI, IFSA*, volume 4529, pages 182 – 190, 2007.

[152] S. Zadrożny et J. Kacprzyk : Bipolar queries : A way to enhance the flexibility of database queries. *In Advances in Data Management, SCI*, pages 49 – 66, 2009.

[153] Z. Zeng, A.K.H. Tung, J. Wang, J. Feng et L. Zhou : Comparing stars : On approximating graph edit distance. *In Proc. of VLDB*, pages 25–36, 2009.

[154] Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng et Lizhu Zhou : Comparing stars : On approximating graph edit distance, 2009.

[155] N. Zhang, T. Özsu, I. Ilyas et A. Aboulnaga : Fix : feature-based indexing technique for xml documents. *In Proc. of VLDB*, pages 259–270, 2006.

[156] Shijie Zhang, Meng Hu et Jiong Yang : Treepi : A novel graph indexing method. *In Proceedings of the International Conference on Data Engineering*, pages 966–975, 2007.

[157] Shu Zhang, Jian Zhong Li, Hong Gao et Zhaonian Zou : A novel approach for efficient supergraph query processing on graph databases. *In Proceedings of the International Conference on Extending Database Technology*, pages 204–215, march 24-26 2009.

[158] Ying Zhang, Houkuan Huang, Dong Yang, Hongke Zhang, Han-Chieh Chao et Yueh-Min Huang : Bring qos to p2p-based semantic service discovery for the universal network. *Personal Ubiquitous Computing*, 13(7):471–477, 2009.

# Annexes

# Annexe A

# Sur la théorie des ensembles flous

Nous définissons dans ce qui suit les notions de base de la théorie des ensembles flous.

## A.1 Notion des ensembles flous

La notion des ensembles flous a été développée par Zadeh [148] en 1965 afin de représenter des classes ou des ensembles dont les limites sont imprécises. Ils permettent de décrire des transitions graduelles entre l'appartenance totale et le rejet. Des exemples typiques de ces classes floues sont celles qui sont décrites à l'aide des adjectifs ou des adverbes de la langue naturelle, comme *pas cher*, *rapide*, *la plupart*, *au moins k*, *presque tous*, etc.

Un ensemble flou $F$ défini sur l'univers $X$ est décrit par une fonction d'appartenance $\mu_F : X \rightarrow [0, 1]$, où $\mu_F(x)$ désigne le **degré d'appartenance** de l'élément $x$ à l'ensemble $F$. Par définition, si $\mu_F(x) = 1$ alors $x$ **appartient complètement** à $F$ ; si $\mu_F(x) = 0$ alors l'élément $x$ **n'appartient pas du tout** à l'ensemble flou $F$ et si $0 < \mu_F(x) < 1$ alors on parle d'une **appartenance partielle** de $x$ à $F$. L'ensemble des éléments de $F$ vérifiant $\mu_F(x) > 0$ représente le **support** de $F$, noté $\{x \in F | \mu_F(x) > 0\}$, et l'ensemble des éléments de $F$ vérifiant $\mu_F(x) = 1$ représente son **noyau**, noté $\{x \in F | \mu_F(x) = 1\}$.

Plus $\mu_F(x)$ est proche de la valeur 1, plus il appartient à $F$. Par conséquent, étant donné $x, y \in F$, on dit que $x$ est préféré à $y$ si et seulement si $\mu_F(x) > \mu_F(y)$. Si $\mu_F(x) = \mu_F(y)$, alors $x$ et $y$ sont de même préférence.

La cardinalité d'un ensemble flou $F$ peut être calculée comme suit : $card(F) = \sum_{x \in F} \mu_F(x)$. Quant à la hauteur d'un ensemble flou $F$, elle correspond au plus grand degré d'appartenance de ses éléments, i.e., $h(F) = \max_{x \in F} \mu_F(x)$. Ainsi, L'ensemble $F$ est dit normalisé si et seulement si $h(F) = 1$.

## A.2 Représentation des ensembles flous

En pratique, la fonction d'appartenance associée à un ensemble flou continue $F$ est généralement représentée par un trapèze (éventuellement ouvertes à gauche ou à droite) ou un triangle [15] défini

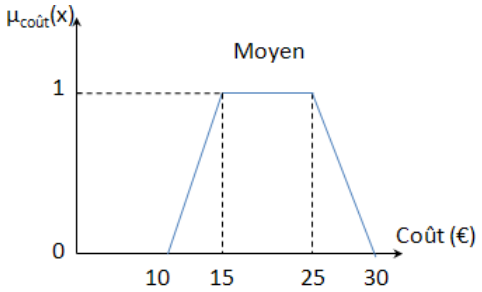



FIGURE A.1 – Fonction d'appartenance de l'ensemble flou des coûts moyens.

par un quadruplet $(\alpha, \beta, \varphi, \psi)$, où $[\alpha, \psi]$ est le support et $[\beta, \varphi]$ est son noyau. Pour assurer la propriété de subjectivité, les valeurs de $\alpha, \beta, \varphi$ et de $\psi$ peuvent être définies par les utilisateurs.

À titre d'exemple, la figure A.1 représente un exemple de fonction d'appartenance des coûts moyens par un trapézoïdale décrit par $(\alpha, \beta, \varphi, \psi)$ où $\alpha = 10€$, $\beta = 15€$, $\varphi = 25€$, $\psi = 30€$. Il est important de noter qu'un degré d'appartenance est interprété comme étant un degré de satisfaction d'un critère (dimension) vis-à-vis de l'utilisateur. Ainsi, la fonction d'appartenance de la figure A.1 permet d'exprimer les degrés de satisfaction de l'utilisateur suivant les valeurs de l'attribut « coût », c'est-à-dire :

- l'utilisateur est complètement satisfait pour un coût appartenant à l'intervalle [15€,25€].
- l'utilisateur n'est pas du tout satisfait pour un coût inférieur ou égale à 10€ou supérieur ou égale à 30€ ;
- et plus le coût est supérieur à 25€ou inférieur à 15€, moins l'utilisateur est satisfait.

Lorsqu'un ensemble flou $F$ est discret, comme celui illustrant le prédicat « foncé » lié aux couleurs [15], il est représenté par $F = \{\mu_F(x_1)/x_1, \ldots, \mu_F(x_n)/x_n\}$. Ceci signifie que l'ensemble $F$ est associé à une fonction d'appartenance de la forme : $\mu_F : \{x_1, \ldots, x_n\} \rightarrow [0, 1]$, où $\mu_F(x_i)$ dénote le degré d'appartenance de l'élément $x_i$ à l'ensemble flou $F$.

## A.3 Intersection et union des ensembles flous

La théorie des ensembles flous offre une panoplie d'opérateurs permettant d'évaluer une combinaison d'ensembles flous.

L'intersection de deux ensembles flous $F$ et $F'$ d'univers $X$ est un ensemble flou ayant une fonction d'appartenance définie par une t-norme, notée $\top$.

Une t-norme ou norme triangulaire $\top$ est une fonction binaire définie par :

$$\begin{array}{rcl} \top : [0,1] \times [0,1] & \rightarrow & [0,1] \\ (x, x') & \mapsto & \top\left(\mu_F(x), \mu_{F'}(x')\right) \end{array}$$

Les opérateurs de type t-norme vérifient les propriétés suivantes :

- associativité : $\top(x, \top(x', x")) = \top(\top(x, x'), x")$,

– commutativité : $\top(x, x') = \top(x', x)$,
– monotonie : $\top(x_1, x'_1) \geq \top(x_2, x'_2)$ si $x_1 \geq x_2$ et $x'_1 \geq x'_2$,
– élément neutre (i.e., l'élément 1) : $\top(1, x) = \top(x, 1) = x$,
– idempotence pour 0 : $\top(0, x) = \top(x, 0) = 0$

où $x, x', x", x_1, x'_1 \in [0, 1]$.

L'union de deux ensembles flous $F$ et $F'$ est également un ensemble flou ayant une fonction d'appartenance définie par une t-conorme, notée $\perp$.

Une t-conorme ou encore appelée conorme triangulaire $\perp$ est une fonction binaire définie comme suit :

$$\begin{array}{rcl} \perp : [0,1] \times [0,1] & \rightarrow & [0,1] \\ \left(x, x'\right) & \mapsto & \perp\left(\mu_F(x), \mu_{F'}(x')\right) \end{array}$$

De la même manière, les opérateurs de type t-conorme vérifient les propriétés suivantes :
– associativité : $\perp(x, \perp(x', x")) = \perp(\perp(x, x'), x")$,
– commutativité : $\perp(x, x') = \perp(x', x)$,
– monotonie : $\perp(x_1, x'_1) \geq \perp(x_2, x'_2)$ si $x_1 \geq x_2$ et $x'_1 \geq x'_2$,
– élément neutre (i.e., l'élément 0) : $\perp(1, x) = \perp(x, 1) = x$,
– idempotence pour 1 : $\perp(1, x) = \perp(x, 1) = 1$,

Les principales normes et conormes triangulaires de la théorie des ensembles flous sont résumées dans le tableau A.1.

TABLE A.1 – Principales normes et conormes triangulaires.

| t-norme $\top$ | t-conorme $\perp$ | Nom |
|---|---|---|
| $\min(x, x')$ | $\max(x, x')$ | Zadeh |
| $x \times x'$ | $x + x' - x \times x'$ | Probabiliste |
| $\max(x + x' - 1, 0)$ | $\min(x + x', 1)$ | Lukasiewicz |
| $\frac{x \times x'}{x' + (1-x') \times (x + x' - x \times x')}$ | $\frac{x + x' - x \times x' - (1-x') \times x \times x'}{1 + (1-x') \times x \times x'}$ | Hamacher |
| $\begin{cases} x & \text{si } x' = 1 \\ x' & \text{si } x = 1 \\ 0 & \text{sinon.} \end{cases}$ | $\begin{cases} x & \text{si } x' = 0 \\ x' & \text{si } x = 0 \\ 1 & \text{sinon.} \end{cases}$ | Weber |

Il a été montré que : (i) la fonction « min » est la plus grande t-norme, c'est-à-dire, pour toute t-norme $\top$, $\top(x, x') \leq \min(x, x')$, $\forall x, x' \in [0, 1]$. Par ailleurs, il a été montré que la fonction « max » est la plus petite t-conorme, c'est-à-dire, pour toute t-conorme $\perp$, $\perp(x, x') \geq \max(x, x')$, $\forall x, x' \in [0, 1]$.