REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

****

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLAB DE BLIDA

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

****

Mémoire de fin d'étude pour l'obtention du diplôme de

Master

Spécialité : **Systèmes Informatiques et Réseaux**

Thème :

Optimisation par Colonies de Fourmis pour l'Extraction des Itemsets à partir de données incertaines

Soutenu publiquement le 03/10/2022 par

CISSE Cheick Fanta Mady

LANYAN Geraldos Audrey Théonas

Devant le jury composé de :

Mme Fareh Messaouda	**Présidente**
Mme Célia HIRECHE	**Examinatrice**
Mme ZAHRA Fatma Zohra	**Promotrice**

Promotion : 2021-2022

Dédicaces

Je dédie ce modeste travail à mon père M. CISSE Adama dont le courage et le dévouement dans l'éducation de ses enfants méritent, à n'en point douter, le plus grand des hommages. Je ne saurai te remercier assez pour le soutien inconditionnel que tu me témoignes depuis toujours.

A ma très chère et aimable mère Mme CISSE Korian DIAKITE grâce à qui je suis ce que je suis aujourd'hui. Ton amour et tes prières ont été, sont et resteront toujours les piliers de mes réussites.

A ma grande sœur chérie Mme TRAORE Kadiatou CISSE et son cher époux Boubakar K TRAORE. Vous êtes d'une manière une mère et un père pour moi. Que le Tout Puissant (Allah) fortifie votre union et vous bénisse.

A mes grands frères et petites sœurs qui sont toujours là pour moi de jour comme de nuit. Je vous aime si fort.

Enfin, je dédie ce travail à tous mes enseignants du primaire à l'université, à mes amis et à toute personne qui a un jour fait partie de ma vie.

CISSE Cheick Fanta Mady.

Je dédie ce travail à mon cher père M. LANYAN Auguste Gratien qui pourrait, si c'est nécessaire, changer la nuit en jour juste pour me voir réussir. Merci Papa pour tout le soutien, les encouragements et efforts que tu as investis en moi. Tu es ma source d'inspiration.

A ma merveilleuse et tendre mère, Mme LANYAN Louise ZOGBLATIN. Le Seigneur m'a béni en faisant de moi l'un de tes fils Maman. Cette molle dédicace ne saurait me permettre de te remercier assez pour ton soutien, tes prières et tes sacrifices. Je suis fier d'être ton fils.

A mes jeunes frères et sœurs. Vos soutiens de tous les jours comptent énormément à mes yeux. Quel bonheur de vous avoir à mes côtés !

A mes amis, proches, collègues et enseignants. Je ne saurai terminer sans vous témoigner ma profonde reconnaissance et gratitude. Vous avez tous contribué d'une manière ou d'une autre à la réalisation de ce travail.

LANYAN Geraldos Audrey Théonas.

Remerciements

Notre premier remerciement est adressé au Tout Puissant Dieu, car sans sa volonté, la réalisation de ce travail serait tout simplement impossible.

Nos sincères remerciements à nos parents et à toutes nos familles respectives qui ont su trouver les mots d'encouragement pour nous soutenir tout au long de la réalisation de ce travail.

Nous tenons à remercier notre promotrice PhD ZAHRA Fatma ZOHRA pour sa supervision. Elle a toute notre reconnaissance et gratitude.

Nous remercions également tous les professeurs de notre université en général et ceux du département d'Informatique en particulier.

Nous désirons aussi remercier nos camarades, nos amis et tous les étudiants étrangers à Blida surtout ceux des communautés béninoise et malienne.

Enfin, nous remercions toutes les personnes de près ou de loin, qui nous ont guidé et consacré de leur temps.

Résumé

L'extraction des itemsets fréquents/hautement utiles consiste en la découverte des items se répétant fréquemment ou présentant une haute utilité dans une base de données. Il s'agit de l'un des problèmes NP-difficiles traité par la majorité des travaux des chercheurs dans ce domaine. Ce problème s'amplifie quand on traite des données incertaines. En effet, nombreuses sont les applications qui produisent chaque jour une multitude de données incertaines.

Dans cette optique, notre travail consiste à traiter l'extraction des itemsets fréquents et/ou hautement utile à partir de données incertaines. Pour ce problème d'optimisation combinatoire, nous nous inspirons de la métaheuristique d'optimisation par colonie de fourmis. En effet cette métaheuristique est basée sur le comportement des fourmis dans la nature.

Après des tests, nous avons remarqué des progrès considérables quant à l'amélioration du temps d'exécution. Cependant, compte tenu du fait qu'il s'agit d'une méthode approchée, la qualité de résultats peut être discutable. Certes, tous les itemsets fréquents et/ou hautement utiles ne sont pas extraits, mais tous les itemsets extraits sont réellement fréquents et de haute utilité.

Mots-clés : extraction des itemsets fréquents, extraction des itemsets hautement utiles, données incertaines, métaheuristiques, optimisation par colonies de fourmis.

Abstract

Frequent/high utility itemset extraction is the discovery of frequently repeating or highly useful items in a database. This is one of the NP-hard problems addressed by the majority of the work of researchers in this area. This problem is amplified when dealing with uncertain data. Indeed, many applications produce a multitude of uncertain data every day.

In this perspective, our work consists in dealing with the extraction of frequent and/or high utility itemsets from uncertain data. For this combinatorial optimization problem, we draw inspiration from the ant colony optimization metaheuristic. Indeed, this metaheuristic is based on the behavior of ants in nature.

After tests, we noticed considerable progress in the improvement of the execution time. However, considering the fact that it is an approximate method, the quality of the results can be questionable. Certainly, not all frequent and/or highly useful itemsets are extracted, but all extracted itemsets are really frequent and of high utility.

Keywords : frequent itemset mining, high utility itemset mining, uncertain data, metaheuristics, ant colony optimization.

ملخص

قاعدة في للغاية مفيدة أو متكررة عناصر اكتشاف من للغاية المفيدة / المتكررة العناصر مجموعات استخراج يتكون عند المشكلة هذه تضخيم يتم .المجال هذا في الباحثين عمل غالبية عالجها التي الصعبة مشاكل من واحدة هذه .بيانات كل المؤكدة غير البيانات من كبيرًا عددًا تنتج التي التطبيقات من العديد هناك ، الواقع في .المؤكدة غير البيانات مع التعامل .يوم

البيانات من للغاية المفيدة أو / و المتكررة العناصر مجموعات استخراج معالجة في عملنا يتمثل ، المنظور هذا في .النمل مستعمرة تحسين metaheuristic التحليل من مستوحون فإننا ، هذه التوافقي التحسين لمشكلة بالنسبة .المؤكدة غير .الطبيعة في النمل سلوك على metaheuristic هذه تستند ، الواقع في

تكون أن يمكن ، تقريبية طريقة لكونها نظرًا ، ذلك ومع .التنفيذ وقت تحسين في كبيرًا تقدمًا لاحظنا ، الاختبار بعد ولكن ، للغاية المفيدة أو / و المتكررة العناصر مجموعات جميع استخراج يتم لا أنه به المسلم من .شك موضع النتائج جودة .عالية فائدة وذات متكررة الواقع في هي المستخرجة العناصر مجموعات جميع

مؤكدة غير بيانات ، للغاية مفيدة مجموعات استخراج ، المتكررة العناصر مجموعات استخراج :المفتاحية الكلمات .النمل مستعمرات بواسطة التحسين ، metaheuristics ،

Table des matières

Liste des tableaux

Liste des figures

Introduction générale

L'Extraction des Itemsets Fréquents (EIF) à partir de données incertaines constitue aujourd'hui l'étape cœur de plusieurs méthodes de fouille de données. De ce fait, elle fait l'objet d'une attention particulière de la part de la communauté des chercheurs dans ce domaine. Une grande partie des travaux proposés dans la littérature dans ce contexte est basée sur des données certaines (données précises). Néanmoins, le monde réel nous procure très souvent des données incertaines, imprécises et en général imparfaites. Cette thèse se confirme par exemple dans le domaine de la médecine lorsque les médecins se trouvent dans l'obligation d'émettre des diagnostics en présence de symptômes imprécis voire incertains. Aussi, les données générées par certains systèmes à base de capteurs sont imparfaites car les capteurs peuvent produire des informations à différents niveaux de confiance. D'où, l'intérêt des chercheurs de la communauté à élaborer plusieurs travaux sur l'extraction des itemsets fréquents à partir de données incertaines.

En outre, il est important de noter que l'extraction des itemsets est un problème NP-Difficile. Ce qui fait souffrir les méthodes exactes d'EIF lors du passage à l'échelle avec un nombre important d'items et un volume de données considérable (dans le cadre du Big Data et des données de masse).

Notre objectif étant de trouver une solution efficace, rapide et moins couteuse en terme de temps d'exécution et de l'espace de recherche, l'étude de quelques métaheuristiques s'avère être d'une nécessité indiscutable puisque nous allons nous inspirer justement d'une métaheuristique afin d'élaborer notre solution.

Les métaheuristiques constituent une famille d'algorithmes inspirés du comportement de certains éléments de la nature. Ces algorithmes sont particulièrement utiles lors de la résolution des problèmes où les algorithmes d'optimisation classiques sont incapables de produire des résultats corrects ou satisfaisants.

La métaheuristique sur laquelle nous nous inspirons est l'algorithme d'optimisation par colonies de fourmis, soit ACO de son nom en version anglaise (Ant Colony Optimization). Comme son nom l'indique, l'algorithme d'optimisation par colonies de fourmis (ACO) s'inspire des comportements des fourmis observés dans la nature ; notamment l'utilisation de la communication indirecte (à travers l'environnement), les traces de phéromone laissées par

les fourmis sur leur passage, et la construction itérative d'une solution globale basée sur une sorte d'intelligence collective.

Chaque fourmi a pour priorité le bien-être de la communauté, ce qui rend leur comportement ''collectif''. Les fourmis obéissent au concept d'Hétérarchie (s'opposant à la Hiérarchie) qui est que chaque individu de la colonie est à priori indépendant et n'est pas supervisé d'une manière ou d'une autre. Chaque individu aide au bon fonctionnement de la communauté et est aidé à son tour par celle-ci dans son évolution. Ainsi, le biais de mécanismes par lequel la colonie est autocontrôlée est relativement simple à étudier.

Dans notre étude, nous allons traiter l'EIF à partir de données incertaines comme étant un problème d'optimisation combinatoire. En effet, le but de notre travail est d'adapter la métaheuristique inspirée du comportement des fourmis (ACO) pour extraire des itemsets à partir de données incertaines. Nous réaliserons pour cela une version séquentielle.

L'ensemble notre travail est réparti en trois grandes étapes. La première partie est sur l'étude de l'existant (ou étude bibliographique). Cette phase consiste à cerner la problématique du sujet. Ce qui nous impose de faire :

- Une étude générale sur les algorithmes classiques d'extraction des itemsets.
- Une étude sur l'extraction des itemsets à partir de données incertaines.
- Une étude sur les métaheuristiques, plus particulièrement sur l'optimisation par colonies de fourmis.
- Une synthèse des travaux faits dans le contexte d'EIF à base des méthodes approchées.

La deuxième partie de notre travail est basée sur la conception de l'application. Nous proposerons dans cette phase une version séquentielle d'une méthode d'EIF à partir de données incertaines basée sur l'ACO.

La troisième et dernière partie de ce travail consiste à implémenter, tester et évaluer la performance et l'efficacité de notre algorithme. Nous en montrerons les limites puis proposerons quelques perspectives.

CHAPITRE I : EXTRACTION D'ITEMSETS FRÉQUENTS À PARTIR DE DONNÉES EXACTES

I. Introduction

Le présent chapitre est consacré à la présentation de quelques concepts fondamentaux, qui sont les piliers de notre recherche. Dans cette perspective, la première rubrique sera donc essentiellement dédiée aux principales notions utiles à l'Extraction des itemsets. Dans la deuxième rubrique, nous parlerons des itemsets fréquents et les règles d'association. La dernière partie concernera les différentes méthodes d'extraction d'itemsets fréquents.

II. Extraction des itemsets

L'idée principale de l'extraction de motifs (itemsets dans notre cas) qui est une méthode du Data Mining consiste à regrouper un ensemble d'item apparaissant simultanément avec une certaine régularité. L'une des premières applications de l'extraction d'itemsets a été l'analyse du panier du consommateur. En effet, les supermarchés se servaient de cette analyse pour mettre en évidence les articles qui étaient souvent achetés ensemble. D'où l'appellation ''itemsets fréquents''.

Prenons un exemple de 12 transactions et 5 items. (Voir Tableau 1)

Tableau 1 : Base de données de transactions

TID	I1	I2	I3	I4	I5
T100	1	0	1	0	0
T200	0	1	0	0	1
T300	0	0	0	1	0
T400	0	1	1	1	1
T500	0	1	1	0	0
T600	0	1	1	0	1
T700	1	1	1	1	0
T800	1	0	1	0	1
T900	1	1	1	0	0
T1000	1	1	1	0	1
T1100	0	1	0	1	0
T1200	1	0	1	0	1

- **Quelques définitions utiles**

Item : Un item est un produit. Dans notre exemple, nous avons 5 produits (items)

Itemset : Un itemset est un ensemble d'item (éléments). Un itemset peut être un singleton.

K-Itemset : est un itemset de k éléments.

Superset : Un superset est un itemset défini par rapport à un autre itemset.

Support : Le support d'un item correspond au nombre de transactions dans lesquels il apparaît. Celui d'un itemset correspond au nombre de transactions dans lesquelles les items apparaissent simultanément.

Itemset fréquent : Un itemset est dit fréquent si son support est supérieur à un seuil minimum défini à l'avance. Les itemsets fréquents sont également susceptibles d'être convertis en règles d'association, ou en d'autres types de règles reposant sur des mesures d'intérêt.

High-utility itemset :(Itemset à haute utilité) Un itemset X est un itemset à haute utilité si son utilité u(X) n'est pas inférieure à un seuil d'utilité minimum (min_util) spécifié par l'utilisateur (c'est-à-dire $u(X) \geq min\_util$). Dans le cas contraire, X est un itemset à faible utilité[1].

Itemset fermé/max : Un itemset p est dit fermé s'il n'existe aucun superset p′ ayant le même support que p et un itemset p est un dit maximal s'il n'existe aucun superset fréquent de p[2].

Générateurs [3]**:** Un itemset A est dit générateur s'il n'existe aucun itemset B tel que $B \subset A$ et Support(B)=Support(A). En d'autres termes, un itemset est générateur si tous ses sous itemsets ont un support strictement supérieur[3].

Itemset multidimensionnel : Les bases de données multidimensionnelles sont destinées à fournir aux décideurs les outils nécessaires pour les aider à la compréhension des données. Cette structure se distingue des données transactionnelles car les ensembles de données contiennent un important volume de données archivées et agrégées définies sur un ensemble de dimensions qui peuvent être organisées en de multiples niveaux de granularité[2].

Itemset multiniveau : La base de transactions peut être traduite en une hiérarchie de concepts. En effet, une hiérarchie de concepts désigne une série de correspondances entre un ensemble de concepts de bas niveau et un autre de plus haut niveau[2].

Itemset rare : Un itemset rare ou peu fréquent est un itemset dont le support est inférieur à un seuil de support minimum spécifié par l'utilisateur.

Itemset négatif : Un itemset négatif est un itemset dont les items présentent un comportement en corrélation négative.

Itemset colossal : Un itemset colossal est un itemset plus robuste dans le sens où si un petit nombre d'éléments sont supprimés de l'itemset, l'itemset résultant aura un support similaire.

Données incertaines : Les données incertaines sont des données dont l'exactitude est incertaine pour des raisons diverses : erreurs de mesure, retard du réseau, etc.

III. Itemsets fréquents et règles d'association

Soient I = {I1, I2, …, Im} un ensemble d'éléments et D, une base de données de transactions où chaque transaction T est un ensemble d'itemsets non vides tel que T ⊆ I. À chaque transaction est associé un identifiant, appelé TID. Soit A un ensemble d'items. Une transaction T est dite contenir A si A ⊆ T. Une règle d'association est une implication de la forme A ⇒ B, où A ⊂ I, B ⊂ I, A ≠∅, B ≠∅, et A∩B = φ. La règle A ⇒ B est valable dans la base de données de transactions D avec support s, où s est le pourcentage de transactions dans D qui contiennent A ∪B. On considère qu'il s'agit de la probabilité, P(A∪B).La règle A ⇒ B a une confiance c dans l'ensemble de transactions D, où c, est le pourcentage de transactions dans D contenant A qui contiennent également B. Ceci est considéré comme la probabilité conditionnelle, P(B|A). C'est-à-dire : support(A⇒B) =P(A∪B) et confiance(A⇒B) =P(B|A)[4].

Les règles qui satisfont à la fois un seuil de support minimum (min sup) et un seuil de confiance minimum (min conf) sont dites fortes. Par convention, nous écrivons les valeurs de support et de confiance de manière à ce qu'elles se situent entre 0% et 100%, plutôt qu'entre 0 et 1,0.

$$\textbf{confiance}(A \Rightarrow B) = P(B|A) = \textbf{support}(A \cup B) / \textbf{support}(A) \qquad (1)$$

On remarquera que la notation P(A∪B) indique la probabilité qu'une transaction contienne l'union des ensembles A et B (c'est-à-dire qu'elle contienne chaque élément de A et B). Elle ne doit pas être confondue avec P(A ou B), qui désigne la probabilité qu'une transaction contienne soit A, soit B[4].

Initialement, les itemsets satisfaisant le support minimum étaient qualifiés de grands. Cependant, ce terme prête quelque peu à confusion car il a des connotations de nombre d'items dans un itemset plutôt que de fréquence d'occurrence de l'ensemble. Par conséquent, nous utilisons le terme ''fréquent'' qui est plus récent. Même si le terme fréquent est préféré à grand, pour des raisons historiques, les k-itemsets fréquents sont toujours désignés par Lk.

L'équation ci-dessus montre que la confiance de la règle A ⇒ B peut être facilement dérivée des supports de A et A∪B. C'est-à-dire qu'une fois que les supports de A, B et A∪B sont trouvés, il est simple de dériver les règles d'association correspondantes A ⇒ B et B ⇒ A et de vérifier si elles sont fortes. Ainsi, le problème de l'exploration des règles d'association peut être réduit à celui de l'exploration d'itemsets fréquents.

Un défi majeur dans l'extraction d'itemsets fréquents à partir d'un grand ensemble de données est le fait qu'une telle extraction génère souvent un nombre énorme d'itemsets satisfaisant le seuil de support minimum (min sup), surtout lorsque min sup est fixé à un niveau bas. En effet, si un itemset est fréquent, chacun de ses sous-ensembles l'est également. Un long itemset fréquent contiendra un nombre combinatoire de sous-itemsets fréquents plus courts[4].

IV. Extraction des itemsets fréquents à partir de données certaines

IV.1 Méthodes séquentielles

Dans cette rubrique, nous étudierons les méthodes d'extraction séquentielles des itemsets fréquents les plus simples.

IV.1.1 L'algorithme Apriori

Apriori[4]est un algorithme novateur développé par R. Agrawal et R. Srikant en 1994 pour l'extraction d'itemsets fréquents pour les règles d'association booléennes. Le nom de l'algorithme tient au fait qu'il utilise une connaissance préalable des propriétés des ensembles fréquents, ce qui sera expliqué plus loin. Apriori utilise une approche itérative connue sous le nom de recherche par niveau, où les k-itemsets sont utilisés pour explorer les (k + 1)-itemsets. Tout d'abord, l'ensemble de 1-itemsets fréquents est trouvé en balayant la base de données pour comptabiliser le nombre d'items, et en rassemblant ceux dont le support est minimal. L'ensemble ainsi obtenu est noté l1. Par la suite, l1 est utilisé pour trouver l2, l'ensemble de 2-itemsets fréquents, qui est utilisé pour trouver l3, et ainsi de suite, jusqu'à ce que plus aucun k-

itemset fréquent ne puisse être découvert. La recherche de chaque Lk nécessite un scan complet de la base de données.

Dans le but d'améliorer l'efficacité de la génération par niveau des itemsets fréquents, une propriété importante dénommée propriété Apriori est appliquée en vue de restreindre le volume de recherche. Cette propriété stipule que tous les sous-ensembles non vides d'un itemset fréquent doivent également être fréquents. Par définition, si un itemset l ne satisfait pas le seuil de support minimum (c'est-à-dire Sup(l) <min_Sup), alors il n'est pas fréquent. Si un item A est ajouté à l'itemset l, alors l'itemset ainsi généré (soit l∪A) ne peut pas être plus fréquent que l. De ce fait, (l∪A) n'est pas fréquent non plus, autrement dit, Sup(l∪A) <min_Sup.

Cette propriété relève d'une catégorie spéciale de propriétés appelée ''Anti-monotonie'' dans le sens où si un ensemble ne peut passer un test, tous ses supersets échoueront pareillement à ce même test. Elle est appelée Anti-monotonie parce que la propriété est monotone dans le cadre de l'échec d'un test.

Comment la propriété d'Apriori est-elle utilisée dans l'algorithme ?

Pour le comprendre, voyons comment Lk-1 est utilisé pour trouver Lk pour $k \geq 2$. Un processus en deux étapes est suivi, composé d'actions join (jonction) et prune (élagage).

- **Etape de jonction :** Pour déterminer Lk, on génère un ensemble de k-itemsets candidats en joignant Lk-1 à lui-même. Nous appellerons cet ensemble de candidats Ck. Considérons l1 et l2 comme des itemsets dans Lk-1. La notation li[j] fait référence au $j^{ème}$ élément de li (par exemple, l1[k-2] fait référence à l'avant-dernier élément de l1). Pour une meilleure application, Apriori suppose que les éléments d'une transaction ou d'un itemset sont triés par ordre lexicographique. Pour le (k-1)-itemset, li, cela implique que les éléments sont triés de telle sorte que li[1] < li[2]< --- < li[k-1]. La jointure, Lk-1*Lk-1, est effectuée, où les membres de Lk-1 sont joignables si leurs premiers (k-2) items sont en commun. La condition l1[k-1] < l2[k-1] garantit tout simplement qu'aucun doublon n'est généré. Les itemsets résultants formés en joignant l1 et l2 sont {l1[1], l1[2], ..., l1[k - 2], l1[k - 1], l2[k - 1]}.
- **Etape d'élagage :**Ck est un superset de Lk, ce qui veut dire que ses membres peuvent être fréquents ou non, mais que tous les k-itemsets fréquents sont inclus dans Ck. Un balayage de la base de données pour déterminer le nombre de chaque candidat dans Ck permettrait de déterminer Lk (c'est-à-dire que tous les candidats dont le nombre

n'est pas inférieur au nombre minimum de support sont fréquents par définition et appartiennent donc à Lk). Ck, cependant, peut être énorme, et donc cela pourrait impliquer des calculs lourds. Pour réduire la taille de Ck, la propriété Apriori est utilisée comme suit. Tous les (k-1)-itemsets qui ne sont pas fréquents ne sauraient être un sous-ensemble d'un k-itemsets fréquents. Par conséquent, si un sous-ensemble (k-1) d'un k-itemset candidats n'est pas dans Lk-1, alors le candidat ne peut pas non plus être fréquent et peut donc être retiré de Ck. Ce test de sous-ensemble peut être effectué rapidement en maintenant un arbre de hachage de tous les itemsets fréquents[4].

- **L'algorithme Apriori** [5]**:**

Apriori (T, t)

Calcul de L1

N ← 2

Tant que LN-1 ensemble vide

CN ← **Apriori** (LN-1)

Pour chaque transaction t ∈ T

Ct ← **Sous-ensemble** (LK, t)

Faire

Pour chaque candidat c ∈ C

Faire

LN ← count ≥ min_Sup

N ←N+1

Retourner LN

Dans cet algorithme les ensembles Ln et Cn sont des enregistrements dans lesquels on stocke les informations et les valeurs des variables. La procédure count permet de calculer et de stocker la fréquence de chaque item de la base de données[5].

- **Avantages :**l'algorithme Apriori est surtout utilisé pour sa précision de découverte des règles d'association pertinentes entre les objets. Il offre également une interprétation

facile des résultats lors de l'extraction des règles d'association, en dépit de leur grand nombre.

- **Inconvénients :**cependant bons nombres d'algorithmes d'extraction relatifs à l'approche support/confiance engendrent une multitude de règles d'association. Un nombre important de configurations d'éléments ne peut générer de règles d'association puisque la recherche de règles d'association exige un temps considérable.

IV.1.2 L'algorithme FP-Growth

L'algorithme[4] Frequent Pattern Growth, ou tout simplement FP-Growth, adopte une stratégie de division et de conquête.

Tout d'abord, l'algorithme comprime la base de données représentant les éléments fréquents en un arbre de motifs fréquents (frequent pattern tree ou FP-tree), qui conserve les informations d'association des itemsets. Il divise ensuite la base de données compressée en un ensemble de bases de données conditionnelles (un type spécial de base de données projetée), chacune associée à un élément fréquent ou "fragment de motif", et exploite chaque base de données séparément. Pour chaque "fragment de motif", seuls les ensembles de données qui lui sont associés doivent être examinés. Par conséquent, cette approche peut réduire de manière substantielle la taille des ensembles de données à rechercher, ainsi que la quantité de données à examiner.

Nous réexaminons l'extraction de la base de données de transactions, D, en utilisant l'approche de FP-growth.

Le premier balayage de la base de données est le même que celui d'Apriori, qui déduit l'ensemble d'item fréquents (1-itemsets) et leur nombre de supports (fréquences). Soit 2 le nombre de support minimum. L'ensemble des items fréquents est trié dans l'ordre décroissant du nombre de supports. L'ensemble ou la liste qui en résulte est désigné par L.

Un FP-tree est ensuite construit comme suit. D'abord, on crée la racine de l'arbre, étiquetée "null". Puis on analyse la base de données une deuxième fois. Les éléments de chaque transaction sont traités dans l'ordre L (c'est-à-dire triés selon le nombre décroissant de supports) et une branche est créée pour chaque transaction. Si deux transactions partagent ensemble un préfixe commun, alors nous incrémentons le compte de support de chaque nœud visité. En vue de faciliter la lecture de FP-tree, une table d'en-tête des items est construite de manière à ce que chaque item pointe vers ses occurrences dans l'arbre via une chaîne de liens

nodaux. De cette manière, le problème de l'exploration de motifs fréquents dans les bases de données est transformé en celui de l'exploration de FP-tree.

Pour terminer, FP-tree est parcouru en créant les sous-fragments conditionnels de base. Pour ce faire, nous extrayons pour chaque fragment de longueur 1 (suffix pattern) tous les préfixes existant dans le chemin du FP-tree (conditional pattern base).

L'itemset fréquent est obtenu en concaténant le suffixe avec les fragments fréquents extraits des FP-tree conditionnels.

Soit D, une base de données de 7 transactions (T1, T2, T3, T4, T5, T6, T7) de 5 items (a, b, c, d, e) avec comme support minimum 2[4].

Tableau 2 : Base de données de transactions

TID	Items
T1	a, b, c
T2	b, c, d
T3	a, c, e
T4	b, d
T5	a, b, c
T6	c, d
T7	a, b, c, e

Tableau 3 : Support des items

Items	Support
c	6
b	5
a	4
d	3
e	2

Tableau 4 :Construction du FP-tree

Items	Support	Lien du Nœud
c	6	
b	5	
a	4	
d	3	
e	2	

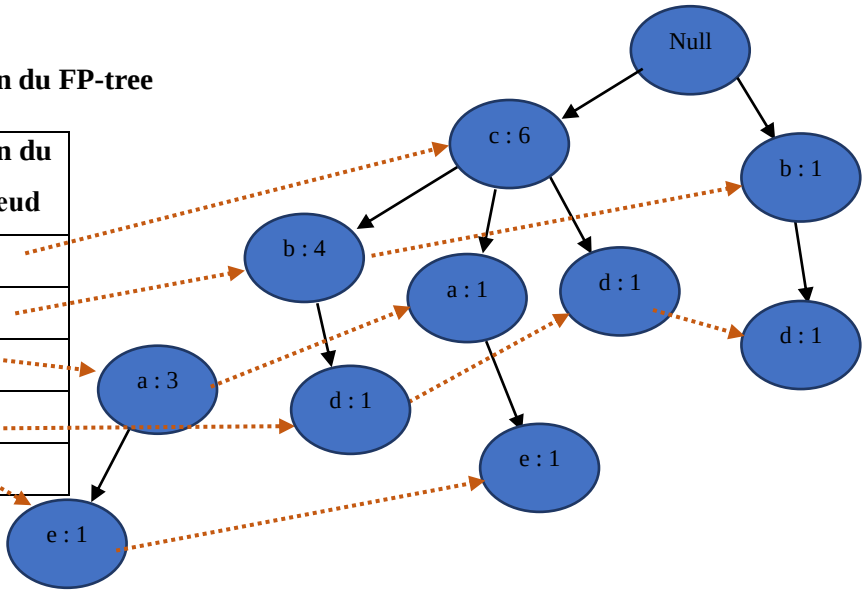


Figure 1 : FP-tree

- **Avantages :** la principale force de l'algorithme FP-Growth réside dans le fait qu'il n'effectue que deux balayages de la base de données des transactions. Un premier balayage pour trouver les k-itemsets et construire la liste des items fréquents et un second pour construire le squelette de l'arbre FP.
- **Inconvénients :**dans bien des cas d'utilisation, nous sommes confrontés à des bases de données transactionnelles bien trop volumineuses, ce qui a pour effet de bloquer le processus de recherche, dans la mesure où les ressources mémoire sont insuffisantes pour accueillir l'ensemble de la structure. La construction de FP-tree peut se révéler très longue en raison de l'utilisation massive de ressources informatiques.

IV.1.3 L'algorithme Eclat

L'algorithme Eclat[6] a été introduit par Zaki.Il consistait à faire une extraction de motifs fréquents en mémoire sans avoir accès au disque. L'algorithme commence par stocker en mémoire une liste d'identifiants de transaction (TID) pour tous les éléments de la base de données. Le support d'un motif I est évalué en croisant les TIDs de chaque élément de I. Eclat lance une recherche d'itemsets fréquents en profondeur dans un premier temps et utilise le concept de classes d'équivalence. Par exemple, ABC et ABD relèvent de la même classe

d'équivalence. Deux k-itemsets appartiennent à la même classe d'équivalence s'ils ont en partage un préfixe de taille (k -1)[6].

NB : Tout comme Eclat, il existe plusieurs autres algorithmes d'extraction d'itemsets fréquents sous format vertical. Parmi les plus célèbres, nous pouvons citer CHARM.

Tableau 5 : tableau comparatif des algorithmes

Algorithmes	Méthodes d'extraction	Caractéristiques
Apriori	Largeur d'abord	i. Nombre considérable d'accès à la base des transactions ii. Complet : pas de perte d'informations iii. Sa performance est proportionnelle à son nombre d'itemsets candidats
FP-Growth	Profondeur d'abord	i. Deux balayages de la base des transactions. ii. Complet iii. Chaque élément fréquent correspond à un chemin dans l'arbre iv. Structure compacte
Eclat	Profondeur d'abord	i. Deux balayages de la base des transactions ii. Rapide : compte les supports des items avec les TIDs

Discussion : Bien que l'algorithme Eclat soit plus rapide que FP-Growth, les deux algorithmes parcourent la base des transactions en profondeur d'abord en effectuant deux balayages. Quant à l'algorithme Apriori, il parcoure la base en largeur d'abord avec une performance proportionnelle à son nombre d'itemsets candidats.

IV.2 Méthodes distribuées et parallèles

Dans cette section, nous présentons les implémentations existantes décrivant les solutions de pointe en matière de traitement distribué et parallèle pour découvrir des itemsets fréquents dans de grands ensembles de données[7].

Tableau 6 : 1er tableau de résumé des approches big data de l'extraction d'itemsets fréquents

Algorithme	Année	Extension	Stratégie de division	Représentation de données
Adaptive-Miner	2018	Apriori	Stratégie de division des données	Horizontal (Filtre Bloom)
CPFPGrowth	2018	FP-Growth	Stratégie de division des données	Horizontal
MR-PFP	2018	Apriori	Stratégie de division des données	Horizontal

Tableau 7 : 2ème tableau de résumé des approches big data de l'extraction d'itemsets fréquents

Algorithme	Pros	Cons	Framework	Phase
Adaptive-Miner	Algorithme dynamique qui change son approche avant chaque itération	Frais généraux de réseau élevés. Nécessité d'analyser l'ensemble des données à chaque itération. L'étape d'intersection prend du temps.	Spark	2P
CPFPGrowth	Stratégie basée sur la mise en cache qui évite le recalcul des données intermédiaires et le tri basé sur les nombres entiers.	Il échoue lorsque la mémoire ne peut pas contenir un FP-tree. Le nombre de partitions doit être plus grand que le nombre d'exécuteurs, mais suffisamment petit pour que le FP-tree puisse tenir dans la mémoire.	Spark	5P
MR-PFP	Analyse d'association spatio-temporelle. Traiter des données de trajectoire à grande échelle comprenant des petits fichiers massifs.	Déséquilibre de la charge de travail.	MapReduce	4P

V. Extraction des itemsets hautement utiles à partir de données certaines

V.1 Introduction à la notion de haute utilité

La méthode d'extraction de données hautement utiles (High Utility Data) a pour but d'extraire des itemsets qui ont une haute utilité. Etant une extension des itemsets fréquents, les itemsets de haute utilité se basent sur la quantité et sur le profit associés à l'ensemble des items. En effet, chaque item possède son propre profit et peut apparaître plus d'une fois dans une même transaction. Ainsi, pour calculer l'utilité d'un itemset, l'on réalise la somme du produit du profit de l'item par son nombre d´apparition dans chaque transaction pertinente. Par conséquent, les itemsets extraits sont des itemsets dont l'utilité est supérieure ou égale à une valeur minimale nommée « min_util ».

Considérons l'exemple suivant :

Tableau 8 : Exemple de transactions et de profits des items

Transactions	Items
T0	a(1), c(1), d(2)
T1	b(1), c(2), d(1), e(4)
T2	a(1), c(1), d(2)
T3	a(2), b(4), c(6) d(3), e(1)
T4	a(1), e(2)

Itemsets	Profits
a	200 Da
b	500 Da
c	1000 Da
d	2000 Da
e	105 Da

Le tableau de gauche représente 5 transactions avec la quantité de chacun des items dans chaque transaction et celui de droite représente les items et leurs profits respectifs.

V.2 Approches du Big Data pour l'extraction de high-utility pattern

La demande en matière de fouille de motifs orientée vers l'utilité a augmenté ces dernières années. L'extraction des itemsets de haute utilité est une tâche essentielle, avec de nombreuses applications à fort potentiel dont le commerce électronique, la finance et l'utilisation en biomédecine. Cependant, la plupart des algorithmes ne sont pas adaptés et ne traitent pas les données à grande échelle. A l'aide d'un tableau, nous allons fournir la description de certaines méthodes d'extraction d'itemsets de haute utilité qui peuvent traiter de grandes données. Les détails des caractéristiques, des avantages et des inconvénients de ces techniques pour l'extraction parallèle d'itemsets de grande utilité sont présentés dans le tableau[7].

Tableau 9 : 1er tableau récapitulatif des approches big data de l'extraction de high-utility pattern

Algorithme	Année	Extension de	Stratégie de division	Représentation de données
BigHUSP	2016	USpan	Stratégie de division des données	Vertical (matrice d'utilité)
EFIM-par	2017	EFIM	Stratégie de division des données	Vertical (Liste d'utilité)
P-FHM+	2018	FHM+	Stratégie de division des données	Vertical (Liste d'utilité)
PHAUIM	2019	HAUI-Miner	Division de l'espace de recherche	Vertical (Liste d'utilité moyenne)
pEFIM	2019	EFIM	Division de l'espace de recherche	Horizontal

Tableau 10 : 2ème tableau récapitulatif des approches big data de l'extraction de high-utility pattern

Algorithme	Pros	Cons	Framework	Phase
BigHUSP	Une stratégie d'élagage efficace évitant de générer des candidats intermédiaires. Extraction de séquences basée sur l'utilité.	Utilise plusieurs travaux de map-reduce. Le temps d'exécution augmente fortement lorsque la taille des séquences d'entrée est doublée.	Spark	4P
EFIM-par	Partition des tâches à l'aide d'un mécanisme de regroupement. Méthode efficace de génération de candidats.	Besoin d'une construction d'arbre efficace. Le mécanisme de regroupement devrait être amélioré.	Spark	3P

P-FHM+	Extraction d'utilité sous contrainte de longueur.	Répartition inefficace de la charge.	Spark	2P

VI. Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'extraction de motifs fréquents, aux différentes méthodes utilisées pour leur extraction et bien sûr aux règles d'association. L'extraction d'itemsets fréquents est aujourd'hui l'un des domaines les plus fondamentaux de la fouille de données qui est largement appliqué dans divers domaines tels que la classification de données, le clustering, l'annotation sémantique, le filtrage collaboratif et la préservation de la vie privée.

Les différentes méthodes précédemment étudiées permettent d'extraire des motifs fréquents à partir de bases de données précises, soit celles dont nous connaissons avec certitude la présence d'éléments dans les transactions. Cependant, il arrive que nous soyons confrontés à un ensemble probabiliste de données incertaines, c'est-à-dire des données dont nous ne sommes pas sûrs de la probabilité de présence des éléments dans les transactions. C'est pourquoi nous consacrerons le prochain chapitre à l'extraction d'items fréquents à partir de données incertaines (probabilistes).

CHAPITRE II : EXTRACTION D'ITEMSETS FREQUENTS À PARTIR DE DONNEES INCERTAINES

I. Introduction

L'extraction d'itemsets fréquents est en principe une méthode de découverte de connaissances. Ces connaissances peuvent se présenter sous la forme d'ensembles d'éléments, d'événements ou d'objets. De plus, l'extraction peut être effectuée soit à partir de bases de données traditionnelles de données précises, soit à partir d'ensembles probabilistes de données incertaines. De ce fait, plusieurs algorithmes ont été développés pour extraire des motifs fréquents à partir de données incertaines.

Dans ce chapitre, nous étudierons les plus connus et les plus pertinents de ces algorithmes.

II. Extraction d'itemsets fréquents incertains

Le but de l'extraction de motifs fréquents à partir de données incertaines est de trouver tous les motifs X satisfaisant la condition expSup(X, D) >= minSup. Une approche probabiliste est utilisée dans cette extraction en raison de l'incertitude de la présence ou de l'absence d'éléments dans la transaction. A la différence des éléments des bases de données transactionnelles, les éléments composant les bases de données incertaines ont en outre leurs propres valeurs de probabilité existentielle. Cette probabilité est notée P(x, tj) représentant la probabilité que l'élément x soit présent dans la transaction tj. A noté que $P(x, tj) \in [0, 1]$.

Une transaction incertaine t est une transaction qui contient des éléments incertains. Une base de données transactionnelles T contenant des transactions incertaines est appelée une base de données de transactions incertaines[8].

III. Définitions

Les imperfections dans les données recouvrent 4 principes qui sont l'imprécision, l'incertitude, l'incomplétude et l'erreur.[9]

Imprécision : Quand les données numériques sont des informations flexibles ou sont mal connues ou encore contiennent des erreurs de mesure.

Incertitude : Lorsque les données numériques ne sont pas déclarées avec une confiance et une garantie absolue.

Incomplétude : Lorsque seule une partie des données réelles est connue.

Erreur : Les informations dont nous disposons sont différentes des véritables informations. On les appelle des données aberrantes et elles constituent le type le plus simple de données imparfaites.

IV. Mesures de calcul de fréquences

Dans les algorithmes d'extraction d'itemsets fréquents à partir de données incertaines, trois mesures sont en effet utilisées pour calculer les fréquences des itemsets. Ce trio de mesures est le suivant : Expected Support, Probabilistic Support et Evidential Support.

IV.1 Expected Support (Support attendu)

Considérons :

n : un ensemble d'éléments ;

X : un sous-ensemble de n tel que X = {x1, x2, …, xk} ;

t_j : la $j^{ième}$ transaction dans la base de données de transactions ;

k : un entier strictement positif et supérieur à 1 ;

P(x, t_j) : la probabilité d'apparition de l'élément x dans la transaction tj ;

- L'expected support de X dans tj noté expSup(X, tj) de k-itemsets est le produit de probabilité de tous les P(xi, tj) pour k transactions[8].

$$\textbf{Autrement dit :} \; expSup(X, t_j) = \prod_{i=1}^{k} P(x_i, t_j) \qquad \textbf{(2)}$$

- L'expected support d'un ensemble probabiliste de n transactions de données incertaines noté expSup(X) correspond à la somme de expSup(X, t_j) sur chaque transaction tj contenantX[8].

$$\textbf{Autrement dit :} \; expSup(X) = \sum_{j=1}^{n} exp\,Sup(X, t_j) \qquad \textbf{(3)}$$

Ainsi, pour généraliser nous aurons :

$$expSup(X) = \sum_{j=1}^{n} \prod_{i=1}^{k} P(x_i, t_j) \qquad \textbf{(4)}$$

Une fois les calculs effectués l'on pourra déterminer si le sous-ensemble X est fréquent ou non. Pour cela, rien de plus de plus simple. X est fréquent si expSup(X) >= minSup, sachant que le minSup est spécifié par l'utilisateur au début[8].

IV.2 Probabilistic Support (Support probabiliste)

Dans les bases de données des transactions incertaines, le support relatif à un ou plusieurs éléments ne peut être représenté par une valeur unique, mais plutôt par une distribution de probabilité discrète.

Appelons T la base de données des transactions et W l'ensemble des éléments possibles dans T, $P_i(X)$ est la probabilité de support d'un itemset X tel que X a un support i[8].

$$P_i(X) = \sum_{w,W,(S(X,wj)=i)} P(w_j) \qquad \textbf{(5)}$$

Où S(X, w_j) est le support de X dans un élément w_j

Soient I l'ensemble de tous les éléments possibles et T, la base de données incertaines où une transaction tj ∈ T est un ensemble des éléments incertains, à savoir, tj ⊆ I.

Pour un itemset X ⊆ tj, en se basant sur l'hypothèse commune que les éléments de X soient indépendants, alors la probabilité existentielle est [8]:

$$P(X \subseteq t_j) = \prod x \in X\, P(x, t_j) \qquad \textbf{(6)}$$

Le support attendu[10] (expected support) de X est alors la somme de la probabilité existentielle sur toutes les transactions[10].

$$expSup(X) = \sum t_j \in T P(X \subseteq t_j) \qquad \textbf{(7)}$$

Le support probabiliste Pi(X) dans une base de données de transactions incertaines T avec des transactions mutuellement indépendantes sachant $0 \leq i \leq |T|$ est calculé comme suit [8]:

$$Pi(X) = \sum_{S \subseteq T, |S|=i} \left(\prod_{t \in S} P(X \subseteq t) \prod_{t \in T-S} (1 - P(X \subseteq t))\right) \qquad \textbf{(8)}$$

Remarques :

- Un itemset X est fréquent probabiliste si son existence dans une transaction minSup est supérieure ou égale au seuil d'utilisateur spécifique minProb[11].

$$P(sup(X) >= minSup) >= minProb \qquad \textbf{(9)}$$

- L'extraction de motifs fréquents probabilistes consiste à trouver tous les itemsets X tel que P(X)>= minProb, ce qui constitue un véritable défi.

Les données sont :

- T : une base de données de transactions incertaines.
- minSup : un seuil minimum de support.
- minProb : un seuil minimum de probabilité fréquente

IV.3 Evidential Support (Support évidentiel)

Une base de données évidentielle appelée EDB en anglais nous permet de stocker des données incertaines et imprécises dont le modèle est basé sur la théorie de Dempster-Shafer. Une telle EDB contient n attributs et d lignes dont chaque attribut i ($1 \leq i \leq n$) dispose d'un domaine que nous noterons Δi de valeurs discrètes.

La fonction de masse élémentaire m d'une hypothèse $X \subseteq \Delta$, dénotée m(X), correspond donc à la fraction du degré de croyance placé sur X et qui n'a pas été distribuée à des sous-ensembles de X. La masse de croyance élémentaire, dénotée bba (bba : basic belief assignment) se définit par[12]:

$$m : 2^{\Delta} \rightarrow [0, 1] \qquad (10)$$

Dans lequel l'espace de puissance 2Ω = {ø {ω1}, {ω2}, {ω1, ω2}, {ω3}, {ω1, ω3}, {ω2, ω3}, {ω1, ω2, ω3}, …, {ω1, …,ωk}} recueille tous les éléments possibles formés à partir des hypothèses et unions d'hypothèses de Ω. On définit les éléments de m de la façon suivante[13]:

$$m_{ij} : 2^{\Delta i} \rightarrow [0, 1] \text{ avec : } m_{ij}(\emptyset) = 0 \text{ et } \sum_{x \subseteq \Delta i} mij(x) = 1 \qquad (11)$$

La fonction de croyance (bel) est définie à partir de la fonction de masse m, la somme des masses de tous les sous-ensembles de A comme suit :

$$bel(A) = \sum_{B \subseteq A} m(B) \qquad (12)$$

Le support d'un itemset évidentiel X dans EDB est sa croyance dans la base[12].

$$sup(X) = bel_{EDB}(X) \qquad (13)$$

Un itemset évidentiel est un itemset dont le support excède le seuil minimum de support. Si l'on considère X comme un itemset évidentiel et σ le produit cartésien des domaines des attributs d'EDB, nous définirons l'ensemble F comme suit [12]:

$$F = \{ X \subseteq \sigma / sup(X) >= minSup \} \qquad (14)$$

V. Les algorithmes d'extraction d'itemsets fréquents à partir des données incertaines

De nos jours[14], plusieurs applications du monde réel sont constamment et régulièrement mises à jour dû à leur productivité massive de données incertaines. Afin de pouvoir traiter les bases de données de ces applications, plusieurs algorithmes furent proposés. Ces algorithmes d'extraction de motifs fréquents à partir des données incertaines peuvent être divisés en deux catégories : les algorithmes basés sur les méthodes de la valeur exacte (exact-based method) et ceux basés sur les méthodes de la borne supérieure (upperbound-based method). Dans les méthodes basées sur la valeur exacte, la probabilité existentielle exacte des items est utilisée pour extraire des motifs fréquents. Et dans celles basées sur la borne supérieure, la borne supérieure de la probabilité existentielle des items est prise en compte.

Nous allons principalement mettre accents sur les algorithmes d'extraction d'itemsets fréquents incertains basés sur les trois algorithmes étudiés dans le chapitre 1. Ces algorithmes sont U-Apriori (basé sur Apriori), UF-Growth (basé sur FP-Growth) et U-Eclat (basé sur Eclat).

V.1 Algorithme U-Apriori

Le U-Apriori a été la toute première méthode conçue pour traiter les problèmes d'extraction d'itemsets à partir des données incertaines. Et la méthode fut élaborée en premier lieu par l'initiateur de l'algorithme Apriori, c'est-à-dire R. Agrawal[15]. L'algorithme incrémente le support attendu (expected support) au lieu du support des motifs candidats. Cette incrémentation est faite par le produit des probabilités existentielles de tous les items $\mathbf{x} \in \mathbf{X}$[16]**.**

Soit MS un expected support minimum que l'algorithme prend en entrée.

La première étape de U-Apriori consiste à scanner la base de données de transactions pour obtenir l'expected support (expSup) de tous les éléments de domaine. Chaque élément présentant un expSup >= MS devient ainsi un motif fréquent cohérent[17].

L'étape 2 consiste à appliquer le processus de Lk-1 pour générer les k-ensembles d'éléments candidats. Ces k itemsets sont ensuite appliqués à la propriété d'Apriori pour élaguer les éléments qui ne sont pas fréquents[17].

Le second scan de la base de données de transactions s'effectue à la troisième étape pour obtenir l'expected support S de chaque k-items candidats. Après une comparaison de S et MS, l'ensemble de k-itemsets fréquents soit LK est formé[17].

L'étape 4 est la génération de tous les sous-ensembles non vides de chaque ensemble d'éléments fréquents[17].

Enfin, l'étape 5 consiste à appliquer la règle ''s ⇒ (l-s)'' pour chaque sous-ensemble de chacun des ensembles fréquents[17].

Hérédité de l'algorithme Apriori, U-Apriori présente de faible efficacité face aux grands ensembles de données. Ce problème de faible efficacité devient encore plus considérable dans des ensembles de données incertaines, en particulier lorsque la plupart des probabilités existentielles sont de faibles valeurs[15].

U-Apriori améliore son efficacité en incorporant la stratégie d'élagage LGS (qui comprend l'élagage local, l'élagage global et le rafistolage en une seule passe). Cette stratégie élimine chaque élément dont la probabilité existentielle est inférieure au seuil d'élagage spécifié par l'utilisateur (qui est local pour chaque élément) de l'ensemble de données probabiliste original D de données incertaines, puis elle extrait les motifs fréquents de l'ensemble de données découpé DTrim résultant[15].

V.2 UF-Growth

Afin d'extraire des motifs fréquents à partir d'ensembles de données probabilistes de données incertaines, un algorithme de minage basé sur un arbre appelé UF-Growth[16] fut proposé par Leung et al. Tout comme son homologue FP-Growth pour l'extraction de données précises, UF-Growth capture le contenu des ensembles de données en construisant également une structure arborescente. Cependant, au lieu de FP-tree, l'algorithme utilise un autre arbre appelé ''UF-tree''. Chaque nœud de cette arborescence se compose d'un item, de la probabilité existentielle et du nombre d'occurrences dans le chemin de cet item. La construction d'un tel arbre se ressemble à celle de FP-tree à l'exception qu'une nouvelle transaction est fusionnée avec un nœud enfant uniquement si le même item et la même probabilité existentielle existent à la fois dans la transaction et dans le nœud enfant. Ce qui réduit considérablement le taux de compression de cet arbre par rapport au taux de compression de FP-tree. Le UF-tree a un nombre limité de nœuds. Ce nombre est la somme

du nombre d'éléments dans toutes les transactions dans la base de données probabiliste de données incertaines.

Pour réduire la consommation de mémoire, l'UF-tree[16] intègre deux techniques d'amélioration. La première technique vise à discrétiser la probabilité existentielle de chaque nœud (par exemple, arrondir la probabilité existentielle à k décimales de sorte que k = 2 décimales), ce qui permet de réduire le nombre éventuellement infini de valeurs de probabilité existentielle possibles à un maximum de 10 à la puissance de k valeurs possibles. La seconde technique d'amélioration est la restriction de la construction des arbres UF aux deux premiers niveaux (c'est-à-dire la construction d'un arbre UF global uniquement pour l'ensemble de données probabilistes original D et d'un arbre UF pour chaque élément fréquent et l'énumération des motifs fréquents pour les niveaux supérieurs (en parcourant les chemins de l'arbre et en décrémentant le nombre d'occurrences) au cours du processus d'exploration.

Par ailleurs, comme les chemins d'un UF-tree[16] ne sont partagés que s'ils ont le même élément et la même probabilité existentielle, l'UF-tree capture avec précision le contenu (notamment les probabilités existentielles) des ensembles de données probabilistes de données incertaines, de sorte que les motifs fréquents puissent être exploités sans produire de faux positifs ou de faux négatifs.

En revanche, l'arbre UF peut être volumineux et ne pas être aussi compact que son homologue FP-tree.

Tableau 11 : Base de transactions

Transactions	Contenus
T1	{a:0.2, b:0.9, c:0.4}
T2	{ a:0.6, b:0.6, c:0.6, d:0.9}
T3	{ a:0.6, b:0.5, d:0.5, e:0.7}
T4	{ a:0.9, b:0.2, c:0.8, e:0.3}

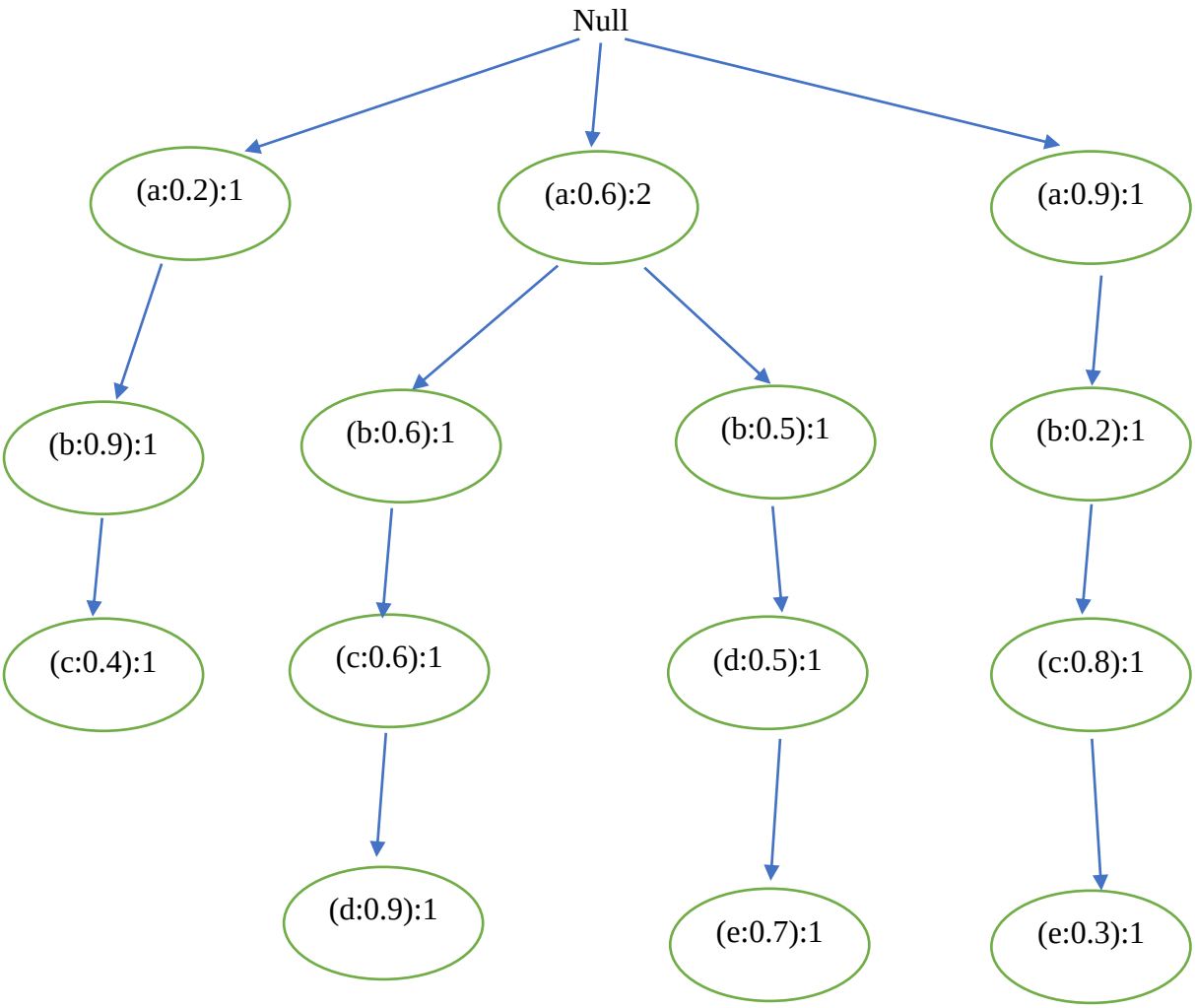


Figure 2 : UF-tree

V.3 U-Eclat

Dans[18] le but d'extraire des motifs fréquents en utilisant la représentation verticale d'ensembles de données probabilistes de données incertaines, Calders et al. ont instancié des "mondes possibles" des ensembles de données en vue de dégager des échantillons instanciés (dans lesquels les données deviennent exactes), puis ont appliqué l'algorithme Eclat à chacun de ces échantillons de base de données instanciés. Le résultat de cet algorithme est baptisé U-Eclat. Compte tenu d'un ensemble de données probabilistes D de données incertaines, U-Eclat génère un nombre aléatoire indépendant r pour chacun des éléments x d'une transaction ti. Si la probabilité existentielle P(x ,ti) de l'élément x dans la transaction ti n'est pas inférieure à un tel nombre aléatoire r (c'est-à-dire que P(x ,ti) ≥ r, alors x est instancié et inclus dans une base de données "Précise" instanciée, qui est ensuite exploitée en utilisant l'algorithme original Eclat. Ce processus d'échantillonnage et d'instanciation se répète donc plusieurs fois et génère par conséquent de multiples bases de données "Précises" échantillonnées. Le support attendu de chaque motif X est le support moyen sur les multiples bases de données échantillonnées. U-Eclat, en tant qu'algorithme basé sur l'échantillonnage, est très efficace mais peu précis. La multiplication des instanciations (c'est-à-dire des échantillons) permet d'améliorer la précision, mais au prix d'une augmentation du temps d'exécution. Et il est important de noter que l'algorithme U-Eclat est un algorithme approximatif[18].

VI. Etudes détaillées des algorithmes d'EIF incertains

Nous allons nous servir des tableaux pour faire dégager certaines caractéristiques des algorithmes d'extraction d'itemsets fréquents basés sur des données incertaines.

VI.1 Paramètres

Le tableau ci-dessous illustre les paramètres généraux des trois algorithmes.

Tableau 12 : Paramètres généraux des algorithmes

Paramètres	***U-Apriori***	***UF-Growth***	***U-Eclat***
Approche	Niveau de profondeur : bas en haut	Par profondeur : diviser et conquérir	Approche verticale des données
Méthode	Générer et tester	Extraction récursive d'arbres	Instanciation des mondes possibles
Scan	Multiple	2	X

VI.2 Avantages et inconvénients

Voyons en grosso modo les avantages et inconvénients des trois algorithmes d'EIF incertains étudiés dans ce chapitre à l'aide d'un tableau.

Tableau 13 : Avantages et inconvénients des algorithmes[10]

Algorithmes	***Avantages***	***Inconvénients***
U-Apriori	Réduit fortement la taille de l'ensemble des candidats.	• La création de DTrim engendre un surcoût. • La base de données est balayée plusieurs fois, ce qui nuit aux performances. • L'efficacité de l'algorithme dépend du pourcentage d'éléments à faible probabilité existentielle.
UF-Growth	Utilise des UF-tree pour extraire des motifs fréquents de données incertaines en ne balayant la base de données incertaine que deux fois.	Chaque item a son propre chemin dans UF-tree.
U-Eclat	• Génère un nombre aléatoire indépendant ''r'' pour chaque élément x de la transaction ti. • Très efficace	• Génération de plusieurs bases de données "précises" échantillonnées. • Moins précis

VI.3 Quelques autres algorithmes d'EIF incertains

Outre les trois algorithmes d'EIF incertains étudiés, plusieurs autres algorithmes ont vu le jour. Ils ont tous pour but d'optimiser plus efficacement l'extraction des itemsets fréquents à partir de données incertaines[14].

Sont les algorithmes d'EIF incertains basés sur :

- Apriori : U-Apriori, UCP-Apriori, MBP, IMBP, etc.
- FP-Growth : UF-Growth, CUF-Growth, CUF-Growth*, ICUF-Growth, UFP-Growth, PUF-Growth, etc.
- Eclat : U-Eclat, UV-Eclat, U-VIPER

- H-Mine : UH-Mine, P-Hmine, CUF-Mine, AT-Mine, CUFP-Mine, etc.
- Lists: UP-List, IUP-List, CUP-List, ICUP-List, LUNA, ILUNA, etc.

Ces algorithmes s'inspirent chacun de leurs précédents et maximisent ainsi leur rentabilité en termes d'efficacité, de précision et de timing[14].

VII. Extraction de motifs fréquents incertains avec contraintes

Les algorithmes basés sur les arbres tels que UF-Growth, UFP-Growth, CUF-Growth, PUF-Growth sont utiles pour trouver tous les modèles fréquents à partir de données probabilistes de données incertaines en général. Cependant, il arrive que les utilisateurs ne soient intéressés que par certains des motifs fréquents. En réponse, Leung et al. ont développé les algorithmes U-FPS et U-FIC en vue d'extraire des ensembles de données probabilistes de données incertaines pour les motifs fréquents qui répondent aux contraintes définies par l'utilisateur. Il s'agit d'extensions de l'algorithme UF-Growth[18].

U-FPS procède d'abord à la vérification des contraintes dans la base de données des transactions incertaines. Ensuite, il établit la structure de UF-tree. Enfin, il explore les motifs fréquents dans l'arbre[18].

U-FIC, quant à lui, utilise les propriétés des contraintes convertibles et organise les éléments du domaine dans l'UF-tree en fonction de certains ordres monotones des valeurs d'attributs pertinents pour les contraintes. De cette manière, l'U-FIC n'a pas besoin d'effectuer de vérification des contraintes contre les extensions de motifs satisfaisant à des contraintes convertibles monotones. De même, U-FIC supprime tous les motifs qui violent toute contrainte convertible anti-monotone[18].

Grâce à l'exploitation des contraintes spécifiées par l'utilisateur, le calcul de U-FPS et U-FIC est proportionnel à la sélectivité des contraintes[18].

VIII. Extraction de motifs fréquents incertains à partir de Big Data (données volumineuses)

Les avancées technologiques font que le volume de données précieuses ne cesse de croître. C'est ainsi qu'est née l'ère du Big Data[18]. Intuitivement, les Big Data sont des données intéressantes dont le volume dépasse la capacité des logiciels conçus pour les capturer, les traiter et les gérer. Cela nous oblige donc à exploiter d'autres formes de traitement de données pour la découverte de connaissances et surtout pour une meilleure prise

de décision. Les chercheurs ont ainsi implémenté MapReduce. Ce dernier est un modèle de programmation de haut niveau capable de traiter de grands volumes de données en utilisant le calcul parallèle et distribué sur de grands clusters ou grilles de nœuds. Deux fonctions clés se retrouvent dans MapReduce : une fonction de mappage et une fonction de réduction[18].

Leung et Hayduk ont proposé l'algorithme MRgrowth qui exploite le modèle MapReduce pour pouvoir extraire des motifs fréquents de grands ensembles de données probabilistes incertaines[18].

Le nœud maître de MapReduce lit et divise un ensemble de données probabilistes D de données incertaines en partitions, puis les assigne à différents nœuds de travail[18].

Comment fonctionne l'algorithme MRgrowth ?

MRgrowth fonctionne en deux étapes. La première consiste à trouver tous les 1-itemsets avec leur support attendu respectif. La deuxième et dernière étape consiste à construire les bases de données projetées appropriées en utilisant FP-tree, CUF-tree ou PUF-tree pour trouver les k-itemsets fréquents (pour k >= 2) avec leur support attendu[18].

IX. Conclusion

Ce chapitre a été consacré exclusivement à l'étude des concepts et des algorithmes des ensembles fréquents incertains. La notion d'incertitude peut être traitée de différentes manières selon que les données soient grandes, contraintes ou évidentes. Le chapitre suivant est consacré à l'étude des métaheuristiques, et par ricochet à l'optimisation par colonies de fourmis.

CHAPITRE III : LES METAHEURISTIQUES

I. Introduction

Dans le domaine de la découverte de connaissances à partir de données incertaines, il est parfois impossible d'obtenir des résultats satisfaisants avec les algorithmes d'optimisation classiques. Afin de compenser ce défaut, plusieurs algorithmes inspirés de la nature ont été développés à ce jour. La famille des métaheuristiques d'optimisation est constituée de ces algorithmes.

Tant en recherche opérationnelle (RO) qu'en mathématiques discrètes ou en informatique, le problème d'optimisation combinatoire représente le principal sujet puisqu'il consiste en la recherche de la solution de meilleure qualité. Les problèmes d'optimisation combinatoire sont caractérisés par leur facilité de définition et leur difficulté de résolution. Ils sont en effet souvent sujets à des problèmes NP-difficiles[19].

Ce chapitre est principalement axé sur la méthode d'optimisation par colonies de fourmis. Suite à la présentation des problèmes d'optimisation combinatoire, nous étudions les méthodes d'optimisation exactes et approchées avec bien sûr quelques algorithmes majeurs de chacune de ces méthodes.

II. Optimisation combinatoire

Les problèmes d'optimisation combinatoire sont une classe des problèmes dont les solutions sont constituées de variables discrètes[19]. Ils sont très généralement résolus par des méthodes dites ''approchées''. Tandis qu'il existe également des problèmes dont les solutions sont des variables à valeurs réelles et qui sont donc résolus par des méthodes dites ''exactes''.

Définition 1 : Un problème d'optimisation combinatoire p(S, f) est défini par un ensemble de variables discrètes X, un ensemble de domaines D et une fonction ''Objectif'' à minimiser (ou à maximiser). Les variables $\{x1, x2, \ldots, xn\} \in X$ sont soumises à des contraintes entre elles. Un ensemble d'affectations $S = \{s=\{(x1, v1), \ldots, (xn, vn)\}$ sachant que $vi \in Di\}$ satisfait toutes les contraintes[20].

S est appelé espace de recherche (solutions) et f fonction de coût ou d'« objectif ». Chacun des éléments de l'ensemble constitue une potentielle solution candidate. Ainsi, une instance I d'un problème de minimisation est un couple (X, f), où $X \subseteq S$ est un ensemble fini de solutions admissibles, et f une fonction de coût à minimiser $f : X \rightarrow R$. Le problème étant maintenant de trouver $s^* \in X$ et que $f(s^*) \leq f(s)$ pour tout $s \in S$[20].

D'une manière similaire, les problèmes de maximisation sont définis en remplaçant tout simplement le ≤ par ≥[20].

∀ s* ∈ S, s* est nommé une solution optimale globale de (S, f) et s ⊆ S est nommé l'ensemble de solutions optimales globales[20].

Définition 2 : La fonction N : S→2s est une structure de voisinage associant à chaque s ∈ S un ensemble de voisinage N(s) ⊆ S sachant que N(s) est appelé voisinage de S[20].

Définition 3 : La notion du minimum local est alors définie grâce à la structure de voisinage[20].

- Une solution s ∈ S est un minimum local relativement à la structure de voisinage N si f(s) ≤ f(s') ∀ s'∈ N(S).
- Une solution s ∈ S est un minimum global si f(s) ≤ f(s') ∀ s'∈ S.
- Optimum global : c'est la valeur de la fonction ''objectif'' de la solution globale. Plusieurs solutions globales peuvent être trouvées mais un seul optimum global est possible.
- Optimum local : il reflète la meilleure solution dans la structure de voisinage

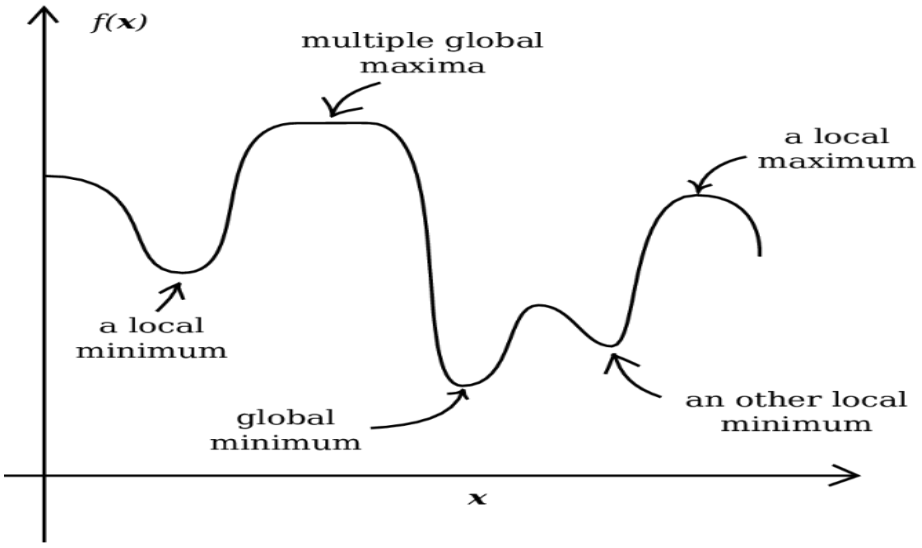


Figure 3: Illustration des notions de minimum local et global[21]

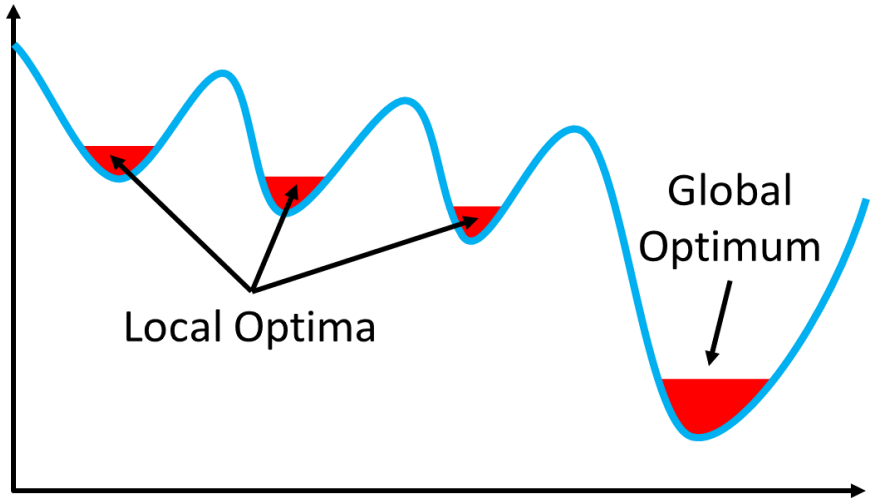


Figure 4: Illustration des notions d'optimum local et global[21]

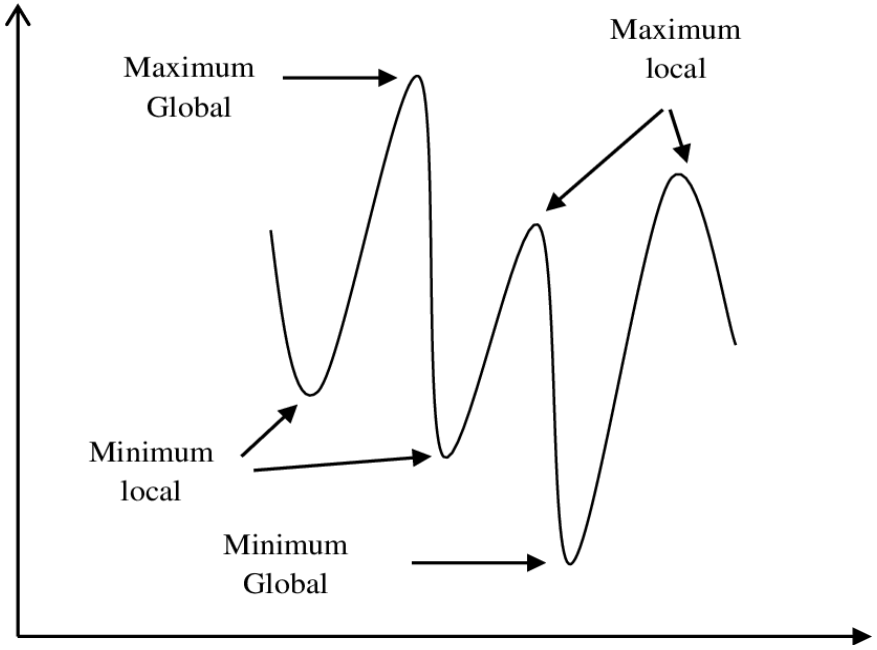


Figure 5 : Illustration des notions de maximum local et global[21]

III. Méthodes d'optimisation exactes

Les algorithmes de résolution de problèmes basés sur les méthodes d'optimisation exactes nécessitent bien souvent une durée de temps déterminée pour pouvoir accomplir leur tâche. Cependant, ce temps peut s'avérer être extrêmement considérable sans compter la gourmandise de ces algorithmes en termes de l'espace mémoire de travail. Les méthodes d'optimisation exactes nécessitent généralement des caractéristiques comme la stricte convexité, la continuité et la dérivabilité de la fonction ''Objectif''[20]. L'intérêt principal des méthodes exactes est la garantie quant à l'optimalité de la solution pour une instance de

problème d'optimisation donné[20]. De ce fait, nous allons étudier 3 méthodes d'optimisation exactes.

III.1 Méthode Branch and Bound (B&B)

Le nom Branch and Bound (B&B)[21] est une appellation anglaise de l'algorithme de séparation et évaluation développé par Land et Doig en 1960. C'est en effet une méthode arborescente de recherche de solution optimale par séparation et évaluation tout comme son nom l'indique. L'algorithme utilise un arbre d'état avec des nœuds et des feuilles pour représenter les états solutions. L'évaluation, la séparation et la stratégie de parcours constituent les 3 principaux axes de la méthode B&B[21].

L'évaluation consiste à évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence. La solution de plus bas coût étant considérée comme la meilleure est alors comparée au coût de chaque nœud parcouru lors de l'exploration. L'exploration de la branche est aussitôt arrêtée dès qu'on trouve un nœud de coût supérieur à celui de la meilleure solution puisque toutes les autres solutions de ladite branche seront nécessairement de coût plus élevé.

La séparation consiste à diviser un problème en des sous-problèmes. Puis un arbre est utilisé pour énumérer toutes les solutions de ces sous-problèmes. Parmi ces solutions seule la meilleure (la solution de coût minimal) est gardée. Ainsi, le problème initial est résolu. Tout nœud de l'arbre non parcouru est appelé nœud actif.

Quant à la stratégie de parcours, nous avons 3 stratégies pour parcourir l'arbre.

La première est la stratégie de la largeur d'abord. Elle effectue moins de séparations du problème initial et favorise les sommets les plus proches de la racine. C'est hélas la stratégie de parcours la moins efficace.

La deuxième stratégie est celle de la profondeur d'abord, qui, contrairement à la première applique plus de séparation au problème initial et favorise les sommets les plus éloignés de la racine. Elle trouve plus rapidement une solution optimale et économise l'espace mémoire.

La troisième et dernière stratégie se nomme la meilleure d'abord. C'est une stratégie qui consiste à explorer les sous-problèmes possédant la meilleure borne tout en évitant d'explorer les sous-problèmes possédant une mauvaise évaluation par rapport à la valeur optimale.

III.2 Méthode Backtracking

Le retour sur trace ou retour arrière[22] (appelé aussi backtracking en anglais) est un algorithme permettant de résoudre des problèmes algorithmiques, notamment de satisfaction de contraintes (optimisation ou décision). Il consiste à sélectionner une variable du problème, et pour chaque affectation possible de cette variable, à tester récursivement si une solution valide peut-être construite à partir de cette affectation partielle. Si aucune solution n'est trouvée, la méthode abandonne et revient sur les affectations qui auraient été faites précédemment (d'où le nom de retour arrière).

En d'autres termes, le retour sur trace est un parcours en profondeur sur l'arbre de décision.

```
procedure EXPLORE(node n)
    if REJECT(n) then return
    if COMPLETE(n) then
        OUTPUT(n)
    for n_i : CHILDREN(n) do EXPLORE(n_i)
```

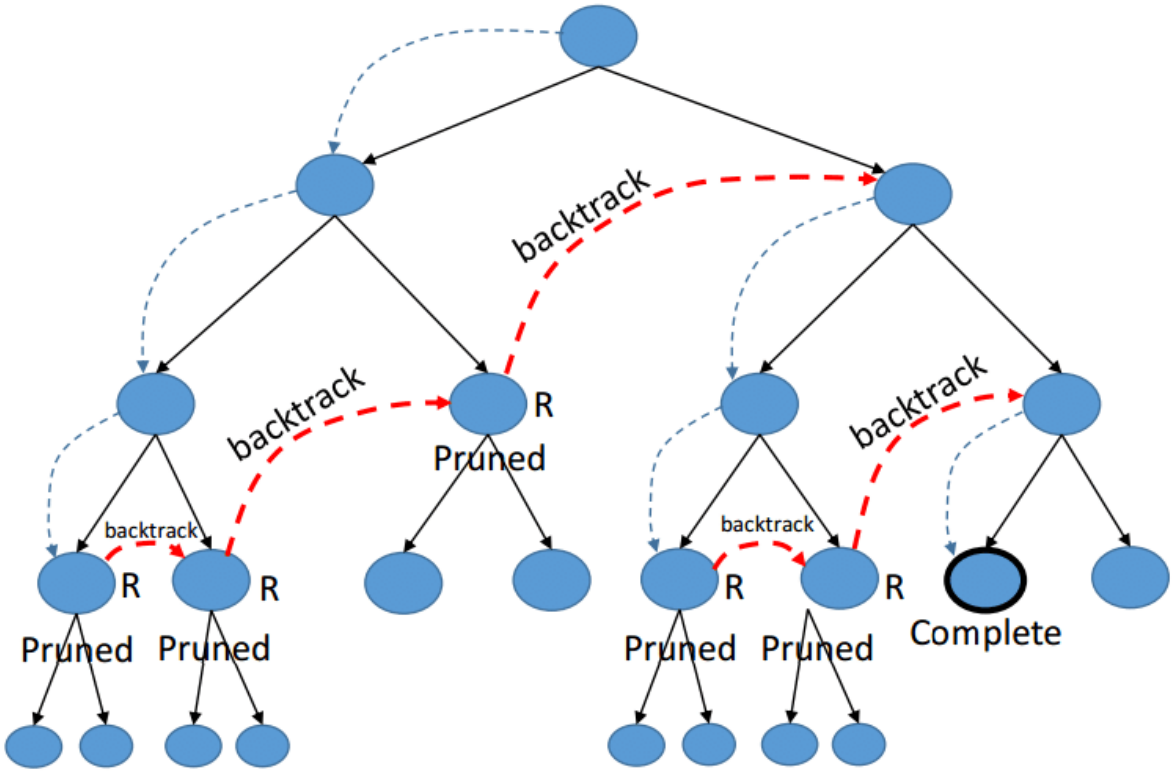


Figure 6 : Arbre de décision de Backtracking.[22]

III.3 Méthode Cutting-Plane

La méthode de coupe plane[23] (ou cutting-plane en anglais) est une méthode utilisée pour trouver une solution entière d'un problème d'optimisation combinatoire (POC) qui se formule sous la forme d'un programme linéaire (PL) : min{cT x : Ax >= b, x ∈ Rn}.

La méthode fut introduite par Ralph E. Gomory, étudiée par Gomory et Vaclav Chvatal puis développée par Schrijver en 1986.

IV. Méthodes d'optimisation approchées

A la différence des méthodes d'optimisation exactes, les méthodes approchées visent à trouver une solution (pas tout à fait complète) de qualité raisonnable et ce, dans un temps de calcul réduit. En somme, les méthodes d'optimisation approchées sont plus efficaces face aux données de grande taille et surtout lorsque l'optimalité de la solution n'est pas une priorité.

IV.1 Heuristiques

L'heuristique ou euristique (du grec ancien εὑρίσκω, heuriskô, « je trouve ») est « l'art d'inventer, de faire des découvertes » en résolvant des problèmes à partir de connaissances incomplètes. Une heuristique est un algorithme conçu spécifiquement pour un problème d'optimisation donné et donc, ne peut être généralisé. Il a pour objectif de trouver dans un temps polynomial une solution réalisable pour ce problème. Le fait qu'il soit propre à un seul problème d'optimisation lui donne une certaine efficacité quant à la rapidité de trouver une solution. Ceci étant, la solution peut ne pas forcement être optimale.

IV.2 Métaheuristiques

Les métaheuristiques constituent une famille d'algorithmes inspirés de la nature dans de nombreux domaines tels que la biologie, la physique, l'éthologie, etc. Ils sont un type d'algorithme aléatoire utilisé pour trouver des solutions optimales. Les métaheuristiques sont des méthodes d'optimisation approchées qui peuvent échapper aux optima locaux et être utilisées dans un large intervalle de problèmes d'ingénierie[24]. On les emploie lorsque les algorithmes d'optimisation classiques sont incapables de produire des résultats satisfaisants.

Nous pouvons classer les métaheuristiques en deux classes à savoir les métaheuristiques à base de solution unique et celles à base de population de solutions.

IV.2.1 Métaheuristiques à solution unique

Ce sont des méthodes qui utilisent une seule solution dans la résolution de problèmes d'optimisation. Voyons quelques algorithmes utiles.

A. Recuit Simulé (Simulated Annealing)

Le recuit simulé[25] (SA) est une méthode de recherche locale évitant les minima locaux. Il a été introduit par Kirkpatrik et al. 1983.Il s'inspire d'une technique des métallurgistes qui faisaient alterner les cycles de réchauffage (de recuit) et de refroidissement afin d'obtenir un alliage parfait.

La méthode consiste à parcourir de manière itérative l'espace des solutions. Une solution S_0 est initialement générée et correspond à l'énergie initiale E_0et à la température initiale T_0 généralement élevée. La solution subit un changement élémentaire à chaque itération de l'algorithme, faisant varier l'énergie du système ΔE. Si la solution trouvée est meilleure que la précédente (c'est-à-dire ΔE négatif) alors cette solution est acceptée. En revanche, si elle est moins bonne que la précédente (c'est-à-dire ΔE est positif) elle sera acceptée avec une probabilité P calculée suivant la distribution de Boltzmann suivante :

$$\mathbf{P = P(E, T) = Exp^{(-\Delta E/T)}} \qquad \mathbf{(15)}$$

En fonction du critère de Metropolis, un nombre $\varepsilon \in [0, 1]$ est comparé à la probabilité P. Si $P \leq \varepsilon$ la nouvelle solution est acceptée. Le fonctionnement du critère de Metropolis est interprété par [25]:

- Si $\Delta E = f(s') - f(s) < 0$ alors $e^{-\Delta E/T} > 1$, donc ε est toujours inférieur à cette valeur, et on accepte la solution s'.
- Si $\Delta > 0$et T est très grande, alors $e^{-\Delta E/T} \approx 1$, tout voisin est systématiquement accepté.
- Si $\Delta > 0$ et T est très petite, alors $e^{-\Delta E/T} \approx 0$, une dégradation a peu de chances d'être acceptée.

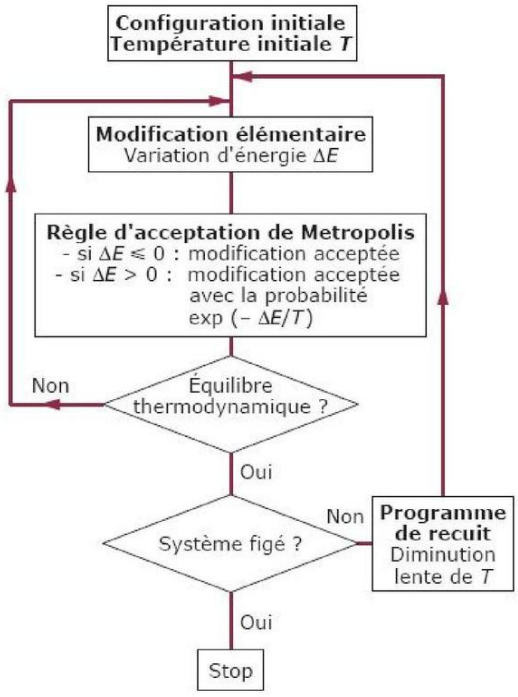


Figure 7 : Organigramme de l'algorithme de recuit simulé.[25]

L'équilibre entre l'intensification et la diversification des solutions dans l'espace de recherche dépend fortement du choix de la température initiale. Et cette dernière dépend de la qualité de la solution initiale S_0. Si S_0 est choisie aléatoirement, on choisit une température initiale relativement élevée en utilisant la règle suivante [25]:

$$T_{k+1} \leftarrow T_k.\alpha \qquad (16)$$

Avec $\alpha \in [0, 1]$, exprimant la diminution de la température de l'itération k à $k + 1$.

B. Recherche Tabou (Tabu Search)

La recherche tabou (TS) est une méthode méta heuristique visant à trouver une solution quasi-optimale à des problèmes d'optimisation combinatoire. A partir d'une solution s dans un ensemble local S de solutions, on génère des sous-ensembles de solutions N(s) voisines de S. La solution acceptée, choisie dans l'ensemble de solutions voisines N(s), est celle qui améliore la valeur de la fonction d'évaluation f[26].

Certaines solutions acceptées par l'algorithme n'améliorent pas vraiment pas la solution courante. Une liste Tabou (Tabu List) T de longueur k dans laquelle sont stockées les k dernières solutions visitées est mise en place. Ainsi, l'on empêche qu'une solution déjà trouvée soit stockée dans la liste Tabou. Par conséquent, la prochaine solution est choisie parmi un ensemble de solutions voisines en dehors des éléments de la liste T. La liste T est

construite selon le principe FIFO, autrement dit, le premier entré est le premier sorti. Lorsque T est pleine, toute nouvelle solution remplace la plus ancienne de la liste[25].

Tableau 14: Tableau comparatif des métaheuristiques à solution unique.[25]

Métaheuristiques	Avantages	Inconvénients
Recuit Simulé (Simulated Annealing)	- Souple vis-à-vis des évolutions du problème et facile à implémenter. - Evite le piège des optima locaux. - Excellents résultats pour un nombre de problèmes complexes	- Nombreux tests nécessaires pour trouver les bons paramètres -Définir les voisinages permettant un calcul efficace de ΔE
Recherche Tabou (Tabu Search)	- Paramétrage simplifié par rapport au recuit simulé.	- Gestion de la mémoire de plus en plus lourde avec des stratégies de mémorisation complexe.

IV.2.2 Les métaheuristiques à populations de solutions

Ces métaheuristiques se servent d'une population de solutions à chaque itération jusqu'à obtenir la solution finale.

A. Algorithme génétique (Genetic Algorithm)

L'algorithme génétique[27] (AG) est une technique d'optimisation et de recherche basée sur la population. Il s'inspire du processus de sélection naturelle et de la génétique. Grâce à son excellente technique d'optimisation des variables, l'AG est souvent utilisé pour résoudre des problèmes complexes[27].

Le principe de fonctionnement de l'AG est le suivant : on part d'une représentation de solutions potentielles (chromosomes) arbitrairement choisies appelées population. Leur performance relative est évaluée avec la fonction d'adaptation (Fitness). Une nouvelle population de solutions est ensuite créée sur la base de ces performances grâce à des opérateurs génétiques simples : la sélection, le croisement et la mutation. Les individus les mieux adaptés sont supposés survivent tandis que d'autres disparaissent ou se reproduisent. Ce cycle est répété jusqu'à l'obtention d'une solution satisfaisante[25].

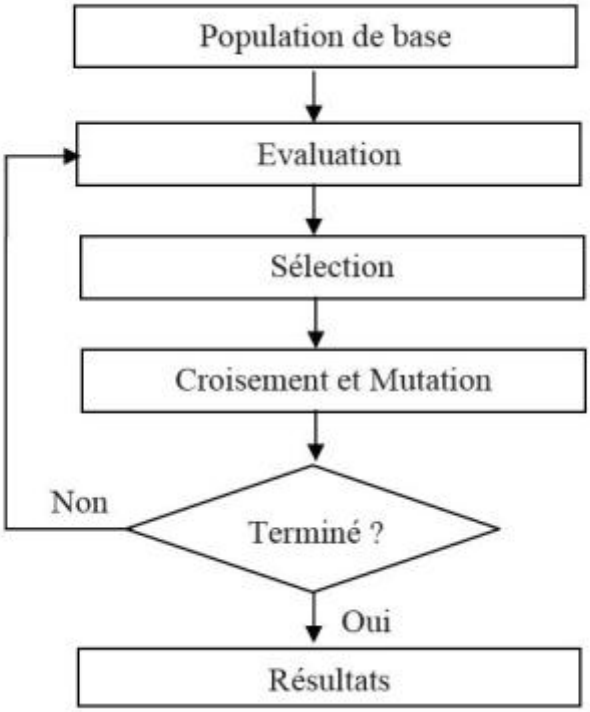


Figure 8 : Fonctionnement de l'algorithme génétique.[25]

B. Optimisation par essaim de particules (Particle Swarm Optimization)

L'optimisation par essaims de particules (OEP)[27]est un algorithme qui imite le comportement social d'espèces collectives telles qu'un banc de poissons, une volée d'oiseaux et un groupe d'humains. Il est également étroitement associé à l'algorithme génétique en raison de leurs nombreuses similitudes. L'algorithme OEP s'intéresse particulièrement à l'exploration et à l'exploitation de l'espace de recherche. Essentiellement, l'exploration est la capacité d'explorer les différentes régions d'un espace de recherche afin de localiser la valeur optimale globale, tandis que l'exploitation est la capacité de concentrer la recherche autour d'une région prometteuse dans l'espoir d'affiner une solution candidate existante. Dans l'optimisation par essaims de particules, une population de solutions candidates, appelées particules, est initialisée sur des positions aléatoires dans un espace de recherche. Au fur et à mesure des itérations, chacune des particules, avec son expérience individuelle et globale, partage des informations avec les autres et converge vers un optimum global[27].

Le déplacement d'une particule est conditionné par les trois types de comportement[25] :

- La particule tend à suivre sa propre voie ;
- La particule tend à revenir vers le meilleur site par lequel elle est déjà passée ;
- La particule tend à se diriger vers le meilleur site déjà atteint par ses voisins.

C. Optimisation par colonies de fourmis (Ant Colony Optimization « ACO »)

L'algorithme ACO est une métaheuristique dans laquelle une colonie de fourmis collabore en vue de trouver une solution satisfaisante à des problèmes d'optimisation difficiles[28]. Cette métaheuristique s'inspire du comportement de recherche de nourriture des fourmis qui, en déposant des phéromones, sont capables d'identifier le chemin le plus court pour transporter leur nourriture[27]. Les algorithmes de colonies de fourmis ont été proposés par Colorni, Dorigo et Maniezzo en 1992 et appliqués au problème du voyageur de commerce pour la toute première fois[29].

Une fourmi artificielle dans ACO est une procédure stochastique qui construit progressivement une solution en ajoutant des éléments appropriés aux solutions en cours de construction. Ainsi, la métaheuristique ACO peut être appliquée aux problèmes d'optimisation combinatoire pour lesquels une heuristique constructive peut être définie[28].

Le problème du voyageur de commerce[28] (PVC) est l'un de ces problèmes, qui consiste à trouver le trajet aller-retour le plus court possible à travers un ensemble donné de villes. Nous pouvons alors imaginer un voyageur de commerce qui doit effectuer son voyage en visitant toutes les X villes une et une seule fois. On peut le représenter par un graphe G = (N, A) avec N l'ensemble des nœuds/villes et A l'ensemble des arcs connectant parfaitement les nœuds. Un poids d_{ij} est attribué à chaque arc (i, j) ∈ A, représentant la distance entre les villes i et j. Le problème PVC tente de trouver un circuit hamiltonien de longueur minimale du graphe. Un circuit hamiltonien est un circuit fermé visitant chaque nœud du graphe une seule fois. Dans notre cas, le circuit hamiltonien détermine la séquence de villes qui minimise la distance que doit parcourir le commerçant[28].

Algorithme 1 [28]**:** Pseudocode de la métaheuristique ACO

Définir des paramètres

Définir les pistes de phéromones

Tant que (condition de clôture n'est pas remplie)

ConstructAntsSolutions

LocalSearch

UpdatePheromones

Faire

L'algorithme 1 montre le pseudocode de base de la plupart des algorithmes ACO, qui consiste en trois procédures principales : ConstructAntsSolutions, LocalSearch et UpdatePheromones.

ConstructAntsSolutions [28]**:**

Chacune des fourmis de la colonie est assignée au hasard à une ville d'où elle se déplace vers une nouvelle ville jusqu'à ce que toutes les villes soient visitées. En retournant à la ville de départ, elle forme un circuit hamiltonien. Afin de décider de la nouvelle ville à visiter, la fourmi artificielle k applique la règle de transition probabiliste suivante qui dépend à la fois des valeurs de phéromone et d'heuristique :

$$\boldsymbol{P_{ij}^{k}} = \frac{[\boldsymbol{\tau_{ij}}]^{\boldsymbol{\alpha}}[\boldsymbol{\eta_{ij}}]^{\boldsymbol{\beta}}}{\sum_{\boldsymbol{u}\in \boldsymbol{N_i^k}}[\boldsymbol{\tau_{iu}}]^{\boldsymbol{\alpha}}[\boldsymbol{\eta_{iu}}]^{\boldsymbol{\beta}}} \qquad \textbf{(17)}$$

Avec, N_i^k l'ensemble des villes qui ne sont pas encore visitées par la fourmi k, τ_{iu} représente la désirabilité de visiter la ville j juste après la ville i, donnée par les pistes de phéromones, et η_{ij}est l'information heuristique.Lorsque l'on résout des problèmes PVC, η_{ij} est habituellement fixé à l'inverse de la distance d_{ij} entre les villes. Les paramètres α et β sont essentiels dans la construction de la solution. Le paramètre α définit la quantité de phéromone des bords qui disparaîtrait à chaque itération. Le paramètre β définit l'importance relative de phéromone par rapport à la valeur heuristique.

LocalSearch [28]**:**

Le pseudocode de base de l'ACO inclut la possibilité d'appliquer une recherche locale permettant d'obtenir un bon compromis entre l'exploration et l'exploitation, qui est au cœur

des métaheuristiques. Une des procédures de recherche locale les plus populaires couplées aux algorithmes ACO est la procédure 3-opt. Fondamentalement, elle supprime trois arêtes dans une tournée, la divisant en trois chemins.

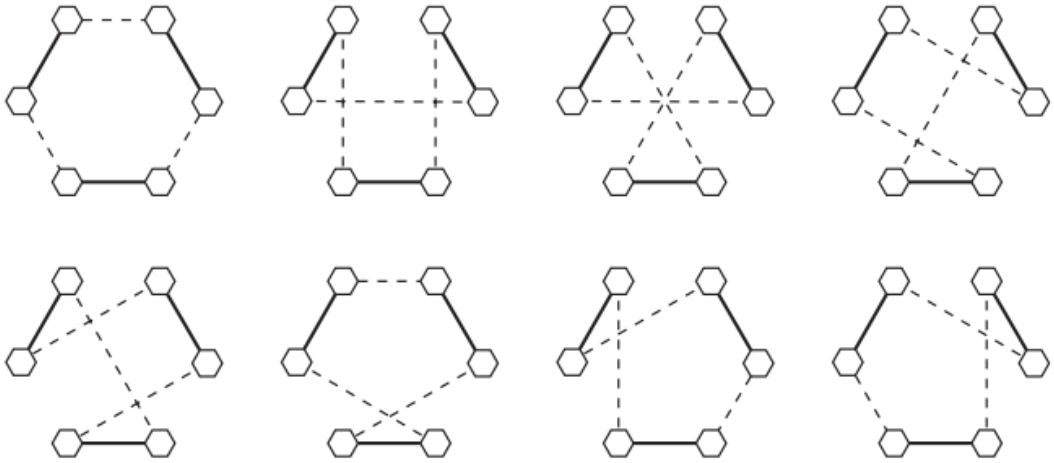


Figure 9 : Application de la recherche locale 3-opt : Possibilités de reconnexion dans une itération de suppression de 3 arrêtes

UpdatePheromones [28]**:**

Dans cette procédure, les pistes de phéromone sont modifiées, augmentant leurs valeurs lorsque les fourmis déposent de la phéromone sur des circuits prometteurs pour guider d'autres fourmis dans la construction de nouvelles solutions, ou diminuant leurs valeurs en raison de l'évaporation de la phéromone pour éviter l'accumulation illimitée de pistes de phéromone et aussi pour permettre d'oublier les mauvais choix.

Le processus d'évaporation empêche l'algorithme de converger prématurément vers des régions sous-optimales et de rester bloqué dans un optimum local, en diminuant τ d'un taux constant ρ (le taux d'évaporation de la phéromone) :

$$\boldsymbol{\tau_{ij} \leftarrow (1-\rho)\tau_{ij}} \qquad \textbf{(18)}$$

Résumé de l'algorithme 1 [28]**:**

ConstructAntsSolutions gère une colonie de fourmis artificielles qui construisent de manière incrémentielle des solutions au problème d'optimisation au moyen de décisions locales stochastiques basées sur les pistes de phéromones et les informations heuristiques. Ensuite, la procédure LocalSearch améliore les tournées des fourmis avec un algorithme de recherche locale optionnel. Enfin, la procédure UpdatePheromones modifie les pistes de phéromone en

fonction de l'évaluation des nouvelles solutions et d'un mécanisme d'évaporation de la phéromone.

Un grand nombre d'extensions de cet algorithme ACO de base peuvent être trouvées dans la littérature. On peut les classer en deux grands groupes. Le premier groupe comprend les propositions qui maintiennent la même procédure de construction des solutions, et introduisent les différences avec l'algorithme de base dans la gestion des pistes de phéromones et la manière dont la mise à jour des phéromones est effectuée. Le second groupe comprend les propositions qui modifient la manière de construire les solutions, comme celles qui modifient plus profondément la structure de l'algorithme de base ou ses caractéristiques.

V. Conclusion

Nous avons étudié au cours de ce troisième chapitre quelques algorithmes clés des méthodes d'optimisation exactes et approchées. Dans bien de domaines requérant l'exactitude des solutions, les méthodes exactes fournissent une solution optimale (complète) mais qui nécessite une durée de calcul élevée quand les données ont une taille importante. Bien que les méthodes approchées tendent à réduire ce temps de calcul, elles ne fournissent pour autant qu'une solution de qualité raisonnable (relativement incomplète). Quant aux méthodes hybrides, elles représentent un compromis des deux précédentes approches.

CHAPITRE IV : APPROCHE PROPOSEE

I. Introduction

Après avoir exposé dans le précédent chapitre les métaheuristiques notables qui sont utilisées pour résoudre les problèmes d'optimisation, nous allons dans ce chapitre présenter notre nouvelle solution d'extraction d'itemsets fréquents à partir de données incertaines en utilisant un algorithme d'optimisation par colonies de fourmis (ACO). Nous appelons cette approche (algorithme) UIM-ACO (Uncertain Itemset Mining based on Ant Colony Optimization).

II. Modélisation UIM en utilisant l'ACO

Dans la mesure où l'extraction des itemsets est, par essence, un problème combinatoire, nous utilisons l'ACO dans cette étude pour la découverte efficace des itemsets fréquents et/ou hautement utiles (High Utility) à partir de données incertaines et exactes.

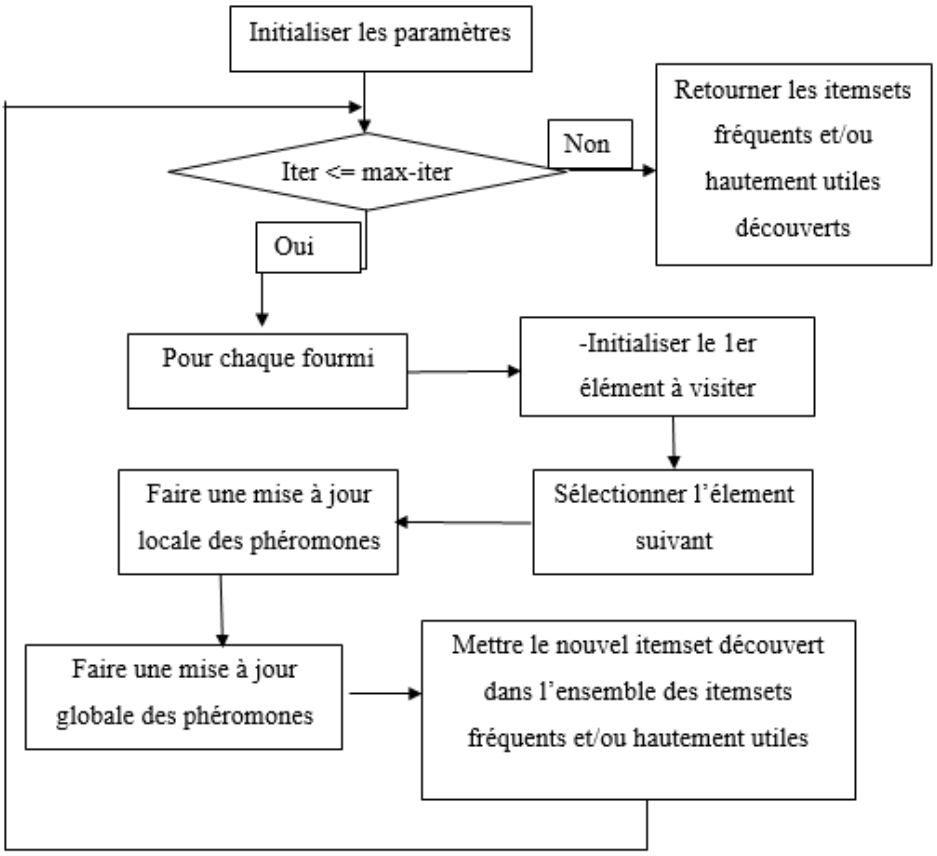


Figure 10 : Organigramme d'extraction des itemsets à partir des données basé sur UIM-ACO

Avec ACO, le problème est abordé par le biais de la simulation d'un certain nombre de fourmis artificielles qui se déplacent sur un graphe, dans lequel chaque sommet correspond à un lieu et chaque arête à une liaison entre deux lieux. À chaque arête est associée une certaine variable appelée phéromone, qui est lisible et modifiable par les fourmis. L'ACO est un algorithme itératif exécuté par **Nb_ants** fourmis. Au cours de chaque itération, chaque fourmi A_t ($1 \leq t \leq$ Nb_ants) construit une solution en se déplaçant de son sommet courant s_c vers le sommet s_d sur le graphe, de sorte que s_d n'ait pas été visité par A_t.

Durant le parcours, le sommet de destination s_d est choisi suivant une probabilité qui est proportionnelle à la valeur de phéromone associée à l'arête (s_c, s_d). Cette valeur de phéromone est ensuite mise à jour pour inciter A_t à construire, dans les prochaines itérations, des solutions similaires aux meilleures solutions déjà construites. Une fois que toutes les fourmis ont atteint le sommet suivant et actualisé les valeurs de phéromone de leur nœud actuel, l'étape de mise à jour globale met à jour la valeur de phéromone du meilleur parcours.

Outre la mise à jour effectuée par chaque fourmi, les valeurs de phéromone de tous les tours passés par les fourmis au cours de cette itération sont également modifiées lors de l'étape de mise à jour globale. La marche vers le nœud suivant, la mise à jour de la phéromone locale et la mise à jour de la phéromone globale sont répétées jusqu'à ce qu'une condition d'arrêt soit satisfaite.

II.1 Problème d'extraction des itemsets

Soit $I = \{i_1, i_2, ..., i_m\}$ un ensemble fini d'items et $X \subseteq I$ un ensemble d'items ; un ensemble d'items contenant k items est appelé un k-itemset. Soit $D = \{T_1, T_2, ..., T_n\}$ une base de données de transactions. Chaque transaction $T_i \in D$ est un sous-ensemble de I.

Nous réalisons l'extraction des itemsets suivant trois différents cas.

A) Extraction d'itemsets fréquents à partir de données exactes

Pour déterminer la fréquence d'un itemset, nous calculons son support dans la base de données, soit le nombre de transactions dans lesquelles apparaît l'itemset.

B) Extraction d'itemsets fréquents à partir de données incertaines

Pour déterminer la fréquence des itemsets, nous utiliserons l'expected support (le support attendu). Le support attendu d'un itemset X dans une transaction T_j noté

expSup(X, T_j) est le produit de la probabilité d'apparition de ses items dans la transaction T_j. Il est défini par :

$$expSup(X, T_j) = \prod_{i=1}^{k} P(x_i, T_j) \qquad (19)$$

avec $P(x_i, T_j)$ la probabilité d'apparition de l'élément x dans la transaction T_j. Le support attendu de l'itemset X dans D est défini comme suit :

$$expSup(X) = \sum_{j=1}^{n} exp\,Sup(X, T_j) \qquad (20)$$

Le support attendu de la transaction T_j est défini comme :

$$expSupT(T_j) = expSup(T_j, T_j) \qquad (21)$$

Un itemset X est dit fréquent si expSup(X) >= min_sup, min_sup étant la valeur minimum de support.

C) Extraction d'itemsets hautement utiles à partir de données incertaines

L'utilité de l'item i_p dans la transaction T_d est définie comme $u(i_p, T_d) = p(i_p) \times q(i_p, T_d)$.

L'utilité de l'itemset X dans la transaction T_d est définie comme : $u(X, T_d) = \sum_{i=1}^{k} u(xi, T_d)$.

L'utilité de l'itemset X dans D est définie comme :

$$u(X) = \sum_{j=1}^{n} u(X, T_j) \qquad (22)$$

II.2 Matrice de phéromones

Nous utilisons une matrice de phéromones pour stocker les valeurs de phéromones d'une paire d'items.

Soit **H** le nombre de 1-itemset et **MP** la matrice de phéromones de taille H * H. Chaque entrée $e_{j,k}$ de la matrice MP correspond à la valeur de phéromone du 2-itemset $i_j i_k$. Les entrées situées le long de la diagonale principale, correspondantes aux valeurs de phéromone d'un item avec lui-même, sont mises à zéro. Pour les autres entrées où $i_j \neq i_k$, $e_{j,k}$ est défini comme suit :

$$e_{j,k} = \begin{cases} expSup(i_j i_k)/2 \; si \; \sum_{i_j i_k \subseteq T_d \wedge T_d \subseteq D} expSupT(T_d) \geq \mathbf{min\_sup} \\ \mathbf{0}, \; sinon \end{cases} \qquad (23)$$

Il ressort de cette équation que $e_{j,k} = e_{k,j}$ alors, MP est une matrice symétrique avec ses entrées symétriques par rapport à la diagonale principale. Par conséquent, nous n'avons besoin de stocker que la moitié de la matrice par rapport à la diagonale principale.

Pour l'étape de mise à jour locale des phéromones, chaque entrée de PM est modifiée selon :

$$\boldsymbol{e_{j,k} = e_{j,k} + expSup(i_j i_k)^{\alpha} / \mathcal{L}} \qquad \textbf{(24)}$$

Avec α($\alpha > 0$), le facteur d'influence local pour ajuster la contribution de la valeur du support attendu dans la phéromone et $\mathcal{L}$, ($\mathcal{L} > 1$) le facteur d'évaporation pour diluer la densité de la phéromone.

Pour la mise à jour globale des phéromones, nous appliquons une approche différente de celle de l'ACO classique. En effet, au lieu de mettre à jour tous les chemins en une seule itération, dès qu'un nouvel itemset fréquent est découvert, notre algorithme ne met à jour que les entrées de ce chemin. Plus précisément, pour chaque entrée $e_{j,k}$ dans le chemin générant un itemset fréquent, la modification suivante est exécutée :

$$\boldsymbol{e_{j,k} = e_{j,k} + \sum_{i_j i_k \subseteq T_d \wedge T_d \subseteq D} expSupT(T_d)^{\beta} / \mathcal{L}} \qquad \textbf{(25)}$$

Avec β ($\beta > 0$), le facteur d'influence global pour ajuster la contribution de la valeur de SET (Somme des expSupT) dans la phéromone.

Dans les équations 2 et 3, la valeur de support attendu est utilisé pour la mise à jour locale de la phéromone, et la valeur SET est utilisée pour la mise à jour globale de la phéromone. Cette différence s'explique principalement par le fait que la granularité de la mise à jour locale est différente de celle de la mise à jour globale. Puisque la mise à jour locale n'est exécutée qu'entre deux sommets adjacents, la valeur exacte de support attendu est utilisée pour mettre à jour la phéromone. Pour la mise à jour globale, toutes les paires de sommets adjacents dans le parcours actuel sont mises à jour, nous utilisons donc SET, l'approximation supérieure du parcours, pour mettre à jour la phéromone.

II.3 Initialisation du chemin de recherche

Lorsqu'une fourmi entame son chemin de recherche, elle sélectionne le premier item i_j proportionnellement, déterminé par :

$$\boldsymbol{P_j = \frac{SET(i_j)^{\gamma}/\mathcal{L}}{\sum_{k=1}^{H} SET(i_k)^{\gamma}/\mathcal{L}}} \qquad \textbf{(26)}$$

Où γ (γ > 0) est le facteur d'influence de la transaction pour ajuster la contribution de la valeur SET dans la sélection du premier item. Pour chaque fourmi At, un quadruplet de chemin de recherche est représenté par AP_t = <i_c, Pa, Succ, TS >, où i_c est l'élément courant visité par At, Pa est l'ensemble des items qui ont été visités par At, Succ est l'ensemble des items n'ont pas été visités par At, et TS est l'ensemble des transactions contenant tous les items de Pa.

II.4 Règle de transition d'état

Une fourmi utilise la règle de transition d'état pour déterminer de manière probabiliste son prochain sommet. Soit At une fourmi. Le prochain sommet représentant l'item i_{next} que At va visiter est déterminé par :

$$\boldsymbol{i_{next} = \begin{cases} argmax_{i_j \in AP_t.Succ\{e_{c,j}\}}, & si\ rand \geq \tau \\ i_k \in AP_t.Succ\ avec\ une\ probabilité\ P_{c,k}, & sinon \end{cases}} \qquad \textbf{(27)}$$

Où $e_{c,j}$ désigne la valeur de phéromone entre $AP_t.i_c$ et i_j dans la matrice de phéromone, rand est un nombre aléatoire uniformément distribué entre 0 et 1, τ ($0 \leq \tau \leq 1$) est le seuil de sélection prédéfini par les utilisateurs, et la probabilité de passage de l'item $AP_t.i_c$ à l'élément i_k est définie comme :

$$\boldsymbol{P_{c,k} = \frac{e_{c,k}}{\sum_{i_l \in AP_t.Succ} e_{c,l}}} \qquad \textbf{(28)}$$

III. L'extraction d'itemsets fréquents à l'aide de l'ACO

III.1 Codification binaire de l'espace de recherche et de transactions

Dans UIM-ACO, nous utilisons représentation binaire des informations des itemsets, une représentation efficace des données relatives aux items dans les algorithmes d'extraction d'itemsets fréquents et hautement utiles, pour identifier les transactions contenant les itemsets cibles. Plus précisément, UIM-ACO utilise une représentation binaire pour les itemsets. Dans une représentation binaire, il y a un bit représentant chaque transaction de la base de données. Si un item i apparaît dans une transaction T_j, alors le bit j de la couverture binaire pour l'item i est mis à un ; sinon, le bit est mis à zéro.

1	0	0	1	0	0	0	0	0	1

Figure 11 : Représentation binaire d'une transaction

Cette transaction représente l'itemset $\{i_0, i_3, i_9\}$

Ce principe peut être étendu aux itemsets. Si X est un itemset, Bin(X) correspond à la représentation binaire de l'ensemble de transactions contenant X. Soit X et Y deux itemsets. Alors, Bin(X ∪ Y) peut être calculé comme Bin(X) ∩ Bin(Y), c'est-à-dire le ET binaire de Bin(X) et Bin(Y).

III.2 Algorithme proposé

L'algorithme 1 décrit notre algorithme UIM-ACO

Algorithme 1 : UIM-ACO

Entrées : Base de transactions **D**, taille de la population **Nb_ants**, valeur de support minimum **min_sup**, facteur d'influence local **α**, facteur d'influence global **β**, le facteur d'évaporation **ℒ**, nombre maximal d'itérations **max_iter**.

Sortie : Itemsets fréquents

---

1. Initialisation () ;
2. **Faire**
3. $F_{new} = Ø$;
4. iter = 1 ;
5. **Tant que** (iter<= max_iter) **faire**
6. **Pour** chaque fourmi A_t **faire**
7. **Si** (iter == 1) alors
8. Initialiser le 1er item visité par A_t en utilisant l'Eq.13 ;
9. Initialiser le quadruplet de chemin de recherche AP_t ;
10. **Fin Si**
11. **Si** (AP_t.Succ≠Ø) alors
12. Sélectionner l'item suivant $i_{t\text{-}next}$ à partir de l'item courant AP_t .i_c en utilisant Eq.14 ;
13. **Si** (SET (AP_t .Pa ∪ $i_{t\text{-}next}$) ≥min_sup AND (AP_t .Pa ∪ $i_{t\text{-}next}$ ∉ SF))alors
14. LocalUpdate (AP_t, $i_{t\text{-}next}$) ;

15. **Si** (expSup(AP_t .Pa) ≥min_sup) alors
16. AP_t .Pa →F_{new} ;
17. GlobalUpdate(AP_t) ;
18. **Fin Si**
19. **Fin Si**
20. **Fin Si**
21. **Fin Pour**
22. iter ++ ;
23. **Fin Tant que**
24. F_{new}→ SF ;
25. **Tant que** (F_{new}≠Ø) ;
26. Afficher tous les itemsets fréquents de SF ;

Dans l'algorithme 1, la procédure Initialisation (), décrite dans l'algorithme 2, est sollicitée à l'étape 1. La boucle principale des étapes 2 à 25 exploite les itemsets fréquents de manière itérative et ce, jusqu'à ce qu'aucune fourmi ne découvre de nouveaux itemsets fréquents. L'étape 3 initialise F_{new}, l'ensemble des itemsets fréquents nouvellement découverts au cours d'une itération, comme étant l'ensemble vide.

L'étape 4 initialise ensuite le nombre d'itérations à 1. La boucle des étapes 5 à 23 traite chaque fourmi individuelle à la condition que le temps de recherche n'ait pas dépassé le seuil maximal. La boucle de l'étape 6 à l'étape 21 décrit les opérations d'une fourmi mettant à jour les valeurs de phéromone et découvrant les itemsets fréquents en une itération. Pour la première itération d'une fourmi, le premier item sélectionné et le quadruplet de chemin de recherche sont initialisés aux étapes 7 à 10. S'il existe des items à explorer, le prochain item à visiter est déterminé à l'étape 12. Si la valeur SET du chemin actuel ajoutant le prochain item nouvellement déterminé n'est pas inférieure au seuil de support minimum, et que le résultat de la fusion n'est pas découvert en tant que itemset fréquent auparavant, l'étape 14 appelle la procédure LocalUpdate () décrite dans l'Algorithme 3.

De même, lorsque le chemin nouvellement formé est un itemset fréquent, la procédure GlobalUpdate(), décrite dans l'algorithme 4, est appelée. L'ensemble de transactions TS dans le quadruplet de chemin de recherche assure que la valeur SET et la valeur de support attendu peuvent être déterminées efficacement en ayant recours aux transactions dans TS, plutôt qu'à la base de données entière. Le temps de recherche est ensuite incrémenté de un. À l'étape 24,

les nouveaux itemsets fréquents découverts au cours de cette itération sont ajoutés à SF, qui est l'ensemble de tous les itemsets fréquents découverts. Enfin, l'étape 26 présente tous les résultats de l'extraction.

Algorithme 2 : Initialisation () ;

1. Balayer la base de données une fois et supprimer les 1-itemsets dont l'expSup≤min_sup ;
2. Initialiser chaque élément de MP en utilisant Eq1 ;
3. SF=**Ø.**

Dans l'algorithme 2, les 1-itemsets de support attendu inférieur à min_sup sont d'abord supprimés à l'étape 1. La valeur de phéromone de chaque paire de 1-itemsets est initialisée à l'étape 2. Enfin, l'ensemble de tous les itemsets fréquents SF est initialisé à l'ensemble vide à l'étape 3.

Algorithme 3 : LocalUpdate (AP_t, $i_{t\text{-}next}$)

1. Mettre à jour la valeur de la phéromone locale sur $e_{c,\ t\text{-}next}$ en utilisant Eq2.
2. $i_{t\text{-}next} \rightarrow AP_t$ .Pa ;
3. AP_t.Succ = AP_t.Succ / $i_{t\text{-}next}$;
4. $AP_t.i_c = i_{t\text{-}next}$;
5. Mettre à jour AP_t.TS.

Dans l'algorithme 3, les valeurs de phéromone correspondant à $AP_t.i_c$ et $i_{t\text{-}next}$ sont mises à jour à l'étape 1. L'étape 2 ajoute $i_{t\text{-}next}$ au chemin de la fourmi actuelle. Puis, le $i_{t\text{-}next}$ est supprimé de l'ensemble des éléments qui n'ont pas été visités à l'étape 3. L'étape 4 utilise le $i_{t\text{-}next}$ comme nouvel élément courant. Enfin, l'ensemble des transactions contenant le chemin actuel est mis à jour à l'étape 5.

Algorithme 4 : GlobalUpdate (AP_t)

1. **Pour** *l*=1 à | AP_t.Pa | - 1 faire
2. Mettre à jour la valeur de phéromone $e_{l, l+1}$ en utilisant Eq3 ;
3. **Fin Pour**

Dans l'algorithme 4, les valeurs de phéromone de chaque paire d'items sur le chemin actuel sont mises à jour séquentiellement de la première paire à la (|AP_t.Pa| - 1)-ième paire, où |AP_t.Pa| est le nombre d'items sur le chemin actuel AP_t.Pa.

IV. Conclusion

Au cours de ce chapitre, nous avons présenté en détail notre algorithme d'extraction d'items fréquents basé sur ACO ou UIM-ACO. Le prochain chapitre se concentrera sur l'implémentation, la validation et les performances de notre algorithme.

CHAPITRE V : TESTS ET VALIDATION

I. Introduction

Dans le chapitre qui précède, nous avons introduit un nouvel algorithme, nommé UIM-ACO, dédié à l'extraction des itemsets fréquents à partir de données incertaines. En effet, notre algorithme fonctionne en trois étapes progressives qui sont :

1. Initialisation du chemin de recherche.
2. Construction itérative des solutions.
3. Mise à jour des valeurs des phéromones.

Dans ce chapitre, nous allons discuter les résultats des essais que nous avons réalisés avec notre approche sur plusieurs bases de test pour en évaluer les performances. Tout d'abord, nous présenterons l'environnement de test de notre approche. Dans un second temps, nous présenterons les bases de tests. Ensuite, nous présenterons notre application ainsi que les résultats.

II. Présentation de l'environnement de travail

Dans cette section, nous présentons l'environnement expérimental qui nous a permis d'évaluer et de tester notre approche d'extraction des itemsets fréquents.

Environnement matériel et logiciel.

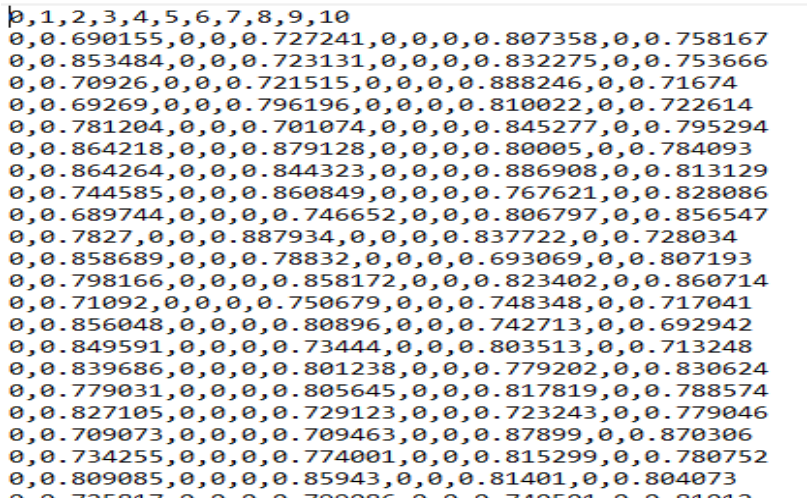
Toutes les expériences ont été réalisées sur un PC équipé d'un processeur AMD RYZEN 5 3500U 2400 MHz et 8 Go de mémoire fonctionnant sur la plateforme Windows 10 Professionnel. Nous avons implémenté notre algorithme UIM-ACO en Python.

III. Bases de tests

Nous avons utilisé des fichiers de données contenant des transactions. Notre algorithme UIM-ACO a ainsi été appliqué sur ces fichiers afin d'extraire des itemsets fréquents.

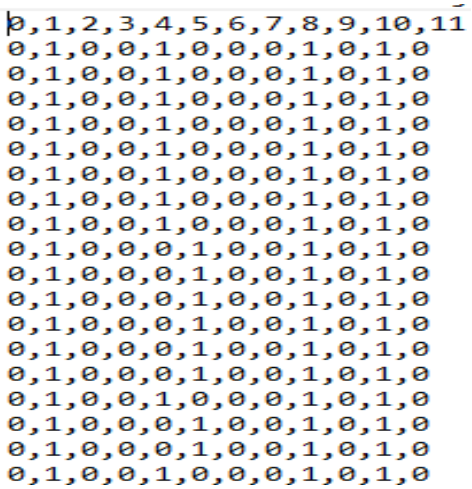
Les deux figures suivantes représentent respectivement des exemples de fichiers de données incertaines et certaines. Chacune des lignes représentant une transaction.

Avec les données incertaines, les items sont représentés par leur probabilité d'apparition tandis qu'avec des données certaines un item est représenté dans une transaction soit par la valeur 1 soit par la valeur 0 (en fonction de sa présence ou de son absence).



```
0,1,2,3,4,5,6,7,8,9,10
0,0.690155,0,0,0.727241,0,0,0,0.807358,0,0.758167
0,0.853484,0,0,0.723131,0,0,0,0.832275,0,0.753666
0,0.70926,0,0,0.721515,0,0,0,0.888246,0,0.71674
0,0.69269,0,0,0.796196,0,0,0,0.810022,0,0.722614
0,0.781204,0,0,0.701074,0,0,0,0.845277,0,0.795294
0,0.864218,0,0,0.879128,0,0,0,0.80005,0,0.784093
0,0.864264,0,0,0.844323,0,0,0,0.886908,0,0.813129
0,0.744585,0,0,0.860849,0,0,0,0.767621,0,0.828086
0,0.689744,0,0,0,0.746652,0,0,0.806797,0,0.856547
0,0.7827,0,0,0.887934,0,0,0,0.837722,0,0.728034
0,0.858689,0,0,0.78832,0,0,0,0.693069,0,0.807193
0,0.798166,0,0,0,0.858172,0,0,0.823402,0,0.860714
0,0.71092,0,0,0,0.750679,0,0,0.748348,0,0.717041
0,0.856048,0,0,0,0.80896,0,0,0.742713,0,0.692942
0,0.849591,0,0,0,0.73444,0,0,0.803513,0,0.713248
0,0.839686,0,0,0,0.801238,0,0,0.779202,0,0.830624
0,0.779031,0,0,0,0.805645,0,0,0.817819,0,0.788574
0,0.827105,0,0,0,0.729123,0,0,0.723243,0,0.779046
0,0.709073,0,0,0,0.709463,0,0,0.87899,0,0.870306
0,0.734255,0,0,0,0.774001,0,0,0.815299,0,0.780752
0,0.809085,0,0,0,0.85943,0,0,0.81401,0,0.804073
```

Figure 12 : Fichier de données incertaines



```
0,1,2,3,4,5,6,7,8,9,10,11
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,0,1,0,0,1,0,1,0
0,1,0,0,1,0,0,0,1,0,1,0
```

Figure 13 : Fichier de données certaines

Dans le fichier de données incertaines nous avons 11 items et dans celui de données certaines 12 items.

III.1 Datasets utilisés

Pour réaliser nos tests, nous avons utilisés quatre datasets de données réelles et semi-synthétiques. Les caractéristiques de ces datasets sont regroupées dans le tableau ci-après :

Tableau 15 : Caractéristiques des datasets utilisés

Dataset	Longueur de la transaction	Nombre d'items	Nombre de transactions	Pourcentage du dataset utilisé
RecordLink	10	29	574913	10%
Skin	4	11	245 058	10%
Mushroom	23	119	8416	100%
Chess	37	75	3196	100%

Ces datasets sont issus de la bibliothèque SPMF Data Mining[30]. Les données incertaines ont été obtenues après affectation d'une probabilité existentielle à chaque item en suivant la loi normale. Les mesures de performances utilisées pour interpréter les résultats sont entre autres le temps d'exécution, les nombre d'itemsets fréquents découverts et la convergence.

III.2 Présentation de l'application

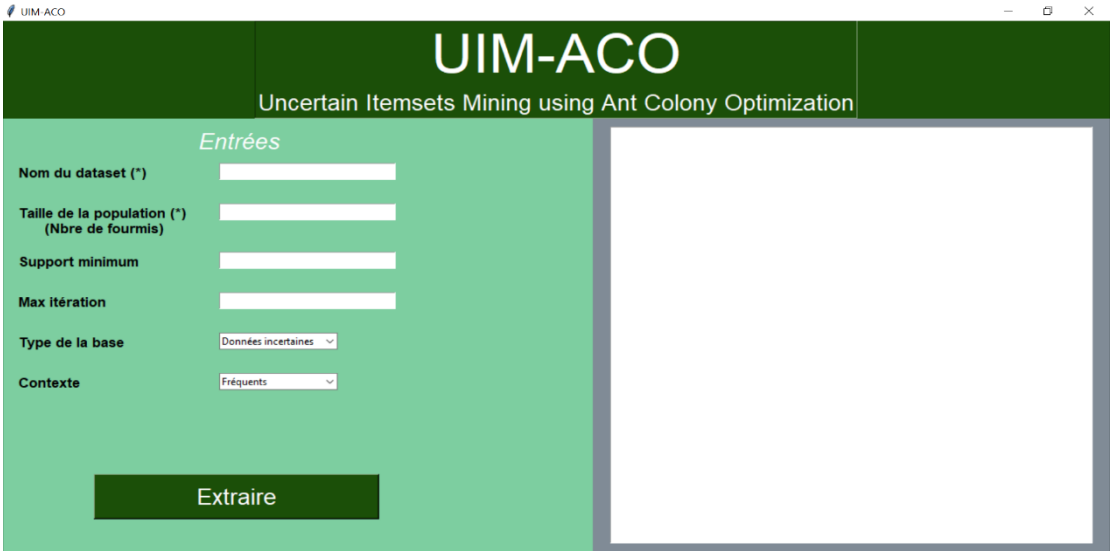


Figure 14 : Interface utilisateur de UIM-ACO

1. Nom du dataset : champ pour saisir le nom du fichier contenant les transactions ;
2. Taille de la population : champ pour saisir le nombre de fourmis ;
3. Support minimum /utilité minimum : champ de saisie du min_sup/min_util ;

4. Max itération : champ de saisie du nombre maximal d'itération ;
5. Type de la base : pour choisir le type des bases de données (certaines ou incertaines)
6. Contexte : pour choisir le contexte d'extraction (Fréquents ou Haute utilité)
7. Extraire : bouton pour lancer l'extraction des itemsets fréquents

Le bouton extraire permet d'extraire les itemsets en fonction du type spécifié puis les résultats sont affichés dans la zone de texte situé à droite de l'interface comme le montre la figure suivante.

IV. Test de performances

Les paramètres répertoriés dans le tableau ci-après sont ceux que nous avons utilisés pour tester notre approche.

Tableau 16 : Paramètres utilisés dans les tests

Paramètres	**Valeur**
Taille de la population	100
Nombre d'exécutions	10
Nombre d'itérations	100

IV.1 Tests sur les données incertaines

IV.1.1 Itemsets fréquents

Nous avons comparé en termes de qualité (nombre d'itemset fréquents) les résultats de notre approche proposée à ceux de l'algorithme de l'algorithme U-Apriori (étant un algorithme exacte) avec différents datasets dans le but de comparer la qualité des résultats. Les tests sont réalisés avec les données incertaines.

Tableau 17 : Pourcentage des itemsets extraits en comparaison avec U-Apriori.

Support minimum	**0.2**	**0.4**	**0.6**
UChess	24.83 %	28.25 %	21.43 %
USkin	5.11 %	5.44 %	6.99 %
URecordLink	11.13 %	24.29 %	23.59 %

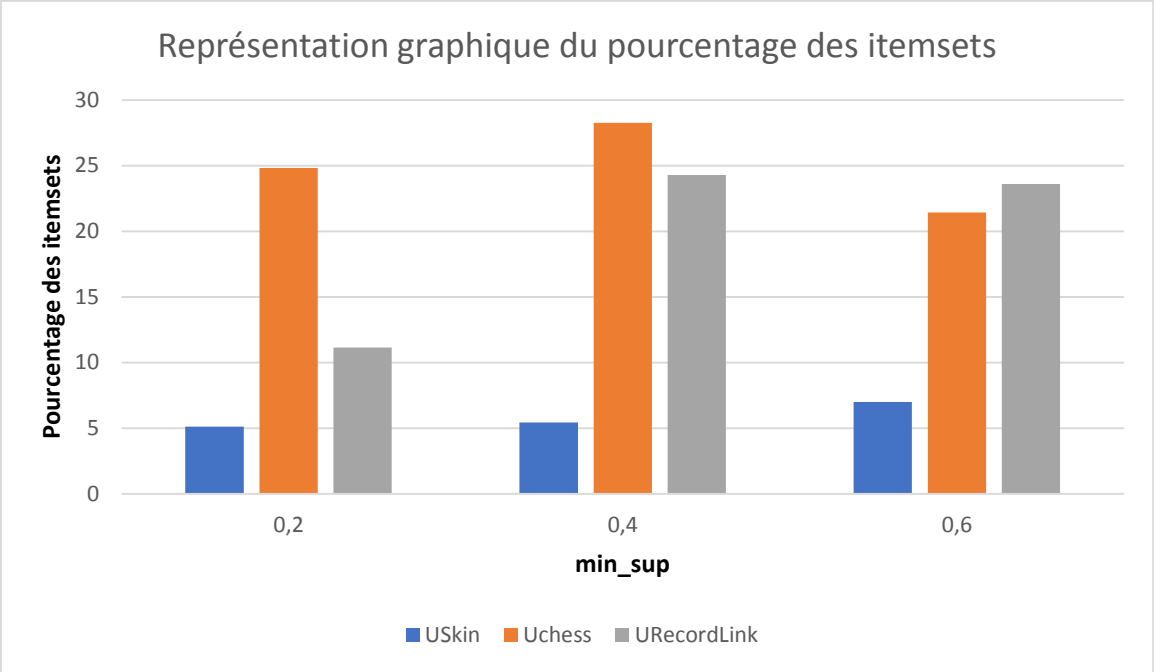


Figure 15 : Comparaison des résultats de UIM-ACO et U-Apriori

Interprétation

On remarque à l'issue de ces tests qu'en comparaison à l'algorithme U-Aprori, la moyenne des itemsets extraits est de 24.84% dans Uchess, 5.85% dans USkin et de 19.67% dans URecordLink.

IV.1.2 Itemsets de haute utilité (High-utility itemsets)

Puisqu'il n'existe aucune méthode approchée traitant l'extraction des itemsets à haute utilité à partir de données incertaines, nous allons évaluer les performances de notre approche sur différents datasets en variant la valeur de l'utilité minimum (min_util).

Tableau 18 : Itemsets de haute utilité découverts dans les différents datasets incertains

Min_util	**0.2**	**0.3**	**0.4**	**0.6**
UChess	41	64	51	75
UMushroom	9	30	33	17
USkin	34	67	68	78

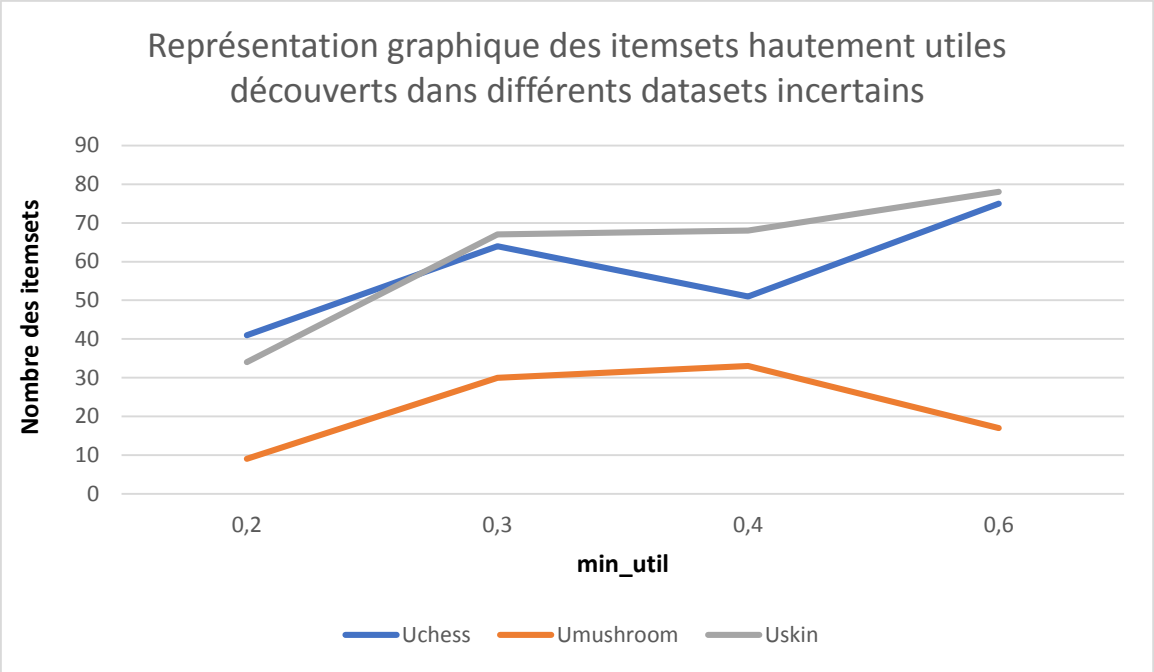


Figure 16 : Itemsets hautement utiles découverts dans différents datasets incertains

Interprétation

Après les tests sur les datasets, nous remarquons une variation des résultats dépendant de la valeur du min_util et du dataset utilisé. Dans USkin, le nombre d'itemsets de haute utilité découverts augmentent au fur et à mesure que min_util augmente. En revanche, dans UMushroom, le nombre d'itemsets diminuent pour un min_util supérieur à 40 %. Enfin, dans UChess, nous observons une croissance pour un min_util entre 20 et 30 % puis entre 40 et 60% mais une baisse est observée entre 30 et 40%.

IV.2 Test sur les données certaines

Ici, nous comparons les résultats de notre approche proposée à ceux de l'algorithme de l'approche basée sur la valeur exacte Apriori avec différents datasets dans le but de comparer la qualité des résultats. Notons que les données de tests sont certaines.

Tableau 19 : Pourcentage des itemsets extraits en comparaison avec Apriori

Support minimum	GAFIM	GA-Apriori	PSOFIM	PSO-Apriori	UIM-ACO (Données certaines)
Skin	63 %	65 %	89 %	91 %	74 %
Mushroom	66 %	70 %	85 %	85 %	65 %

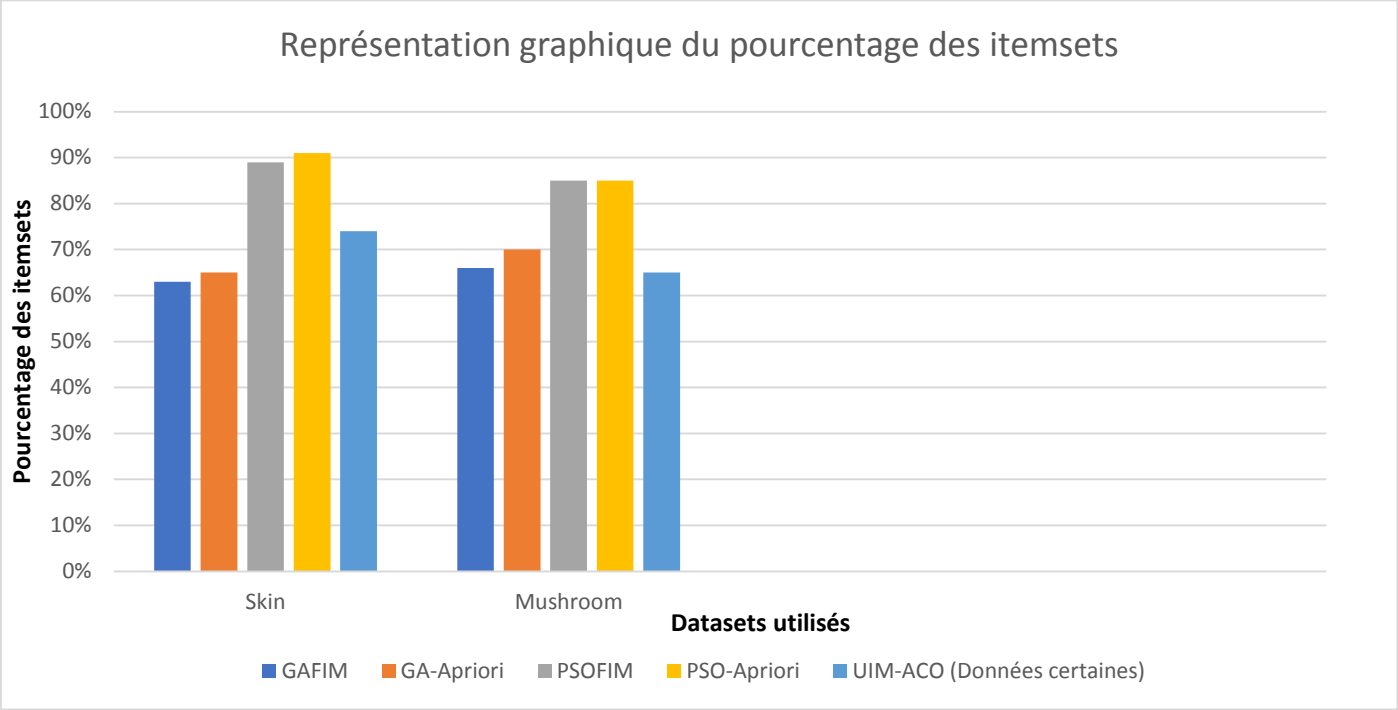


Figure 17 : Comparaison des résultats de plusieurs algo. Et Apriori

Interprétation

Nous remarquons après ces tests que notre algorithme (UIM-ACO) est beaucoup plus performant avec les données certaines qu'avec les données incertaines. Aussi, il est plus efficace que l'algorithme GAFIM d'une moyenne de 10% et presque autant performant que l'algorithme GA-Apriori. Cependant, il est moins efficace que les algorithmes PSOFIM et PSO-Apriori.

V. Conclusion

Ce chapitre a été consacré à la présentation de notre environnement d'implémentation et de test ainsi que les bases de données transactionnelles que nous avons utilisé pour nos expérimentations. En outre, nous avons présenté notre interface puis nous avons exposé les résultats de nos expériences. Les tests de performance que nous avons effectués permettent de découvrir la relation entre le temps d'exécution d'UIM-ACO et certaines variables nécessaires à l'extraction des itemsets fréquents.

Conclusion générale

Dans ce mémoire, nous nous sommes attelés à l'extraction des itemsets fréquents dans les bases de données transactionnelles constituées de données incertaines. Dans ce sens, nous avons proposé une nouvelle approche pour l'extraction des itemsets fréquents. En effet, nous avons commencé ce travail en présentant les notions préliminaires liées à l'extraction des itemsets. Nous avons également décrit les méthodes séquentielles et parallèles pour l'extraction des itemsets fréquents. Ensuite, dans le deuxième chapitre, nous avons étudié l'extraction des itemsets fréquents à partir de données incertaines, les mesures de calcul de fréquence ainsi que les algorithmes d'extraction d'itemsets fréquents incertains. Nous avons également effectué une étude détaillée de ces principaux algorithmes. Notre nouvelle approche appelée UIM-ACO a été conçue et implémentée pour effectuer l'extraction des itemsets fréquents à partir de bases de données certaines et incertaines. La principale originalité de notre approche réside dans le fait qu'elle est une première dans le cadre de l'extraction d'itemsets fréquents à partir de données incertaines de masse. De plus, notre approche pourrait être adaptée à tout type d'algorithme d'EIF incertains. Afin d'améliorer la mise en œuvre et la flexibilité de notre approche, voici quelques perspectives qui nous semblent intéressantes :

- l'utilisation de données entièrement réelles pour évaluer la pertinence des résultats obtenus ;
- compte tenu de la croissance exponentielle des bases de données dans le monde réel, la mise en œuvre d'une méthode parallèle distribuée est nécessaire pour améliorer la scalabilité ;
- Optimiser notre algorithme pour réduire le temps d'exécution, notamment lors de l'utilisation de datasets contenant un nombre très important de données.
- Améliorer notre approche de manière générale afin qu'elle soit plus précise tant pour l'extraction des itemsets fréquents que pour l'extraction des itemsets de haute utilité.
- Améliorer notre algorithme afin qu'il puisse extraire des itemsets à partir de données évidentielles.

Bibliographie

[1] P. Fournier, V. Jerry, C.-W. Lin, R. Nkambou, B. Vo, and V. S. Tseng Editors, *High-Utility Pattern Mining Theory, Algorithms and Applications*. 2018.

[2] J. Han, M. Kamber, and J. Pei, "Advanced Pattern Mining," in *Data Mining*, Elsevier, 2012, pp. 279–325. doi: 10.1016/b978-0-12-381479-1.00007-1.

[3] B. Nabila, mémoire de master, "Optimisation par colonies d'abeilles pour l'extraction des itemsets fréquents à partir de données évidentielles," Saad Dahlab Blida, 2019.

[4] J. Han, M. Kamber, and J. Pei, "Mining Frequent Patterns, Associations, and Correlations: Basic Concepts and Methods," in *Data Mining*, Elsevier, 2012, pp. 243–278. doi: 10.1016/b978-0-12-381479-1.00006-x.

[5] A. NOUASRIA, mémoire de master, "EXTRACTION D'ASSOCIATIONS LEXICALES FORTES DANS LES COMMENTAIRES," UNIVERSITÉ DU QUÉBEC, 2016.

[6] O. Khemiri, mémoire de master, "Proposition d'une nouvelle approche d'extraction des motifs fermés fréquents," UNIVERSITÉ DE TUNIS EL MANAR, 2018.

[7] S. Kumar and K. K. Mohbey, "A review on big data based parallel and distributed approaches of pattern mining," *Journal of King Saud University - Computer and Information Sciences*. King Saud bin Abdulaziz University, May 2019. doi: 10.1016/j.jksuci.2019.09.006.

[8] A. Cuzzocrea, C. K. Leung, and R. Kyle, "Approximation to expected support of frequent itemsets in mining probabilistic sets of uncertain data," *Procedia - Procedia Comput. Sci.*, vol. 60, pp. 613–622, 2015, doi: 10.1016/j.procs.2015.08.195.

[9] C. K. Chui, B. Kao, and E. Hung, "Mining Frequent Itemsets from Uncertain Data," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4426 LNAI, pp. 47–58, 2007, doi: 10.1007/978-3-540-71701-0_8.

[10] A. Sara, mémoire de master, "Extraction des itemsets fréquents incertains en utilisant l'apprentissage profond," Université Saad Dahlab Blida, 2019.

[11] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng, "Mining uncertain data with probabilistic guarantees," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*,

pp. 273–282, 2010, doi: 10.1145/1835804.1835841.

[12] B. LIAUDET, “Cours de Data Mining 3-Modelisation”, EPF - 4ème année - IAP

[13] Purvi Y. Rana; Pragna Makwana; Kishori Shekokar, “A Brief Survey on Frequent Patterns Mining of Uncertain Data”, International Journal of Engineering Sciences & Research Technology (IJESRT), Vol.3, No. 12, 2012-12-30.

[14] R. Davashi, “ILUNA : Single-pass incremental method for uncertain frequent pattern mining without false positives,” *Inf. Sci. (Ny).*, vol. 564, pp. 1–26, 2021, doi: 10.1016/j.ins.2021.02.067.

[15] E. H. Chun-Kit Chui, Ben Kao, “Mining Frequent Itemsets from Uncertain Data,” The University of Hong Kong, 2007.

[16] L. Yue, “Review of Algorithm for Mining Frequent Patterns from Uncertain Data,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 15, no. 6, pp. 17–21, 2015.

[17] T. Calders, C. Garboni, and B. Goethals, “Efficient Pattern Mining of Uncertain Data with Sampling,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6118 LNAI, no. PART 1, pp. 480–487, 2010, doi: 10.1007/978-3-642-13657-3_51.

[18] C. C. Aggarwal and J. Han, *Frequent pattern mining*, vol. 9783319078. Springer International Publishing, 2014. doi: 10.1007/978-3-319-07821-2.

[19] J.-K. Hao, M. Habib, and P. Galinier, “Métaheuristiques pour l’optimisation combinatoire et l’affectation sous contraintes,” *Rev. d’Intelligence Artif.*, 1999.

[20] A. Abdiya, “Application des techniques des métaheuristiques pour l’optimisation de la tâche de la classification de la fouille de données,” Université des sciences et de la technologie d'Oran Mohamed Boudiaf, 2012.

[21] A. H. Land and A. G. Doig, “An Automatic Method for Solving Discrete Programming Problems,” *50 Years Integer Program. 1958-2008 From Early Years to State-of-the-Art*, pp. 105–132, 2010, doi: 10.1007/978-3-540-68279-0_5.

[22] Verenich, Ilya & Nguyen, Hoang & La Rosa, Marcello & Dumas, Marlon. (2017). White-Box Prediction of Process Performance Indicators via Flow Analysis. 10.1145/3084100.3084110.

[23] C. Alexander Schrijver. John Wiley & Sons, *Theory of linear and integer programming*, vol. 36, no. 8. Pergamon, 1998. doi: 10.1016/S0898-1221(98)91141-5.

[24] D. Oliva, E. H. Houssein, and S. Hinojosa, *Metaheuristics in Machine Learning: Theory and Applications*. Studies in Computational Intelligence, 2021. doi: 10.1007/978-3-030-70542-8_31.

[25] H. L. Sidi Mohamed Douiri, Souad Elbernoussi, "Cours des Méthodes de Résolution Exactes Heuristiques et Métaheuristiques." , Université Mohamed V, Facuté des Sciences de Rabat

[26] F. Glover and M. Laguna, "Tabu Search," in *Handbook of Combinatorial Optimization: Volume1--3*, D.-Z. Du and P. M. Pardalos, Eds. Boston, MA: Springer US, 1998, pp. 2093–2229. doi: 10.1007/978-1-4613-0303-9_33.

[27] C. K. Teoh, A. Wibowo, and M. S. Ngadiman, "Review of state of the art for metaheuristic techniques in Academic Scheduling Problems," *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 1–21, 2015, doi: 10.1007/s10462-013-9399-6.

[28] P. González, R. R. Osorio, X. C. Pardo, J. R. Banga, and R. Doallo, "An efficient ant colony optimization framework for HPC environments," *Appl. Soft Comput.*, vol. 114, Jan. 2022, doi: 10.1016/j.asoc.2021.108058.

[29] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System : Optimization by a Colony of Cooperating Agents," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 26, no. 1, 1996.

[30] "SPMF: A Java Open-Source Data Mining Library." https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php (accessed Sep. 29, 2022).