REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU

FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire présenté pour l'obtention

Du diplôme de Master

Intitulé

Etat de l'art sur les règles d'associations séquentielles : proposition d'une nouvelle solution PSBI

- *Proposé et encadré par : Mr AIT EL HADJ ALI*
- *Présenté par : ALLIK LYNDA BELLADJEL OUIZA*

- **Promotion : 2018/2019**

Remerciement

On tient à exprimer nos vifs remerciements à ALLAH qui nous
A donné la patience afin de finir ce mémoire malgré les
Difficultés rencontrées.
On souhaite d'abord exprimer nos vive reconnaissance pour notre
Promoteur
*Monsieur **AIT EL HADj** qui a bien voulu*
Accepter l'encadrement de ce mémoire et il nous a toujours guidé sans
Jamais nous contraindre,
On le remercie très sincèrement pour nous 'avoir donné l'opportunité de
mener à bien ce
Travail par sa disponibilité et son soutien, pour le temps qu'il nous 'a
Consacré et ses nombreux et précieux conseils.
Je souhaite également remercier les membres du jury pour nous 'avoir fait
L'honneur d'accepter de juger et d'évaluer notre travail.
Ces remerciements ne seraient pas complets si on ne citait pas le département
d'informatique
De l'université MOULOUD MAMMERI qui nous a permis de réaliser notre rêve de
continuer nos études en ce domaine très intéressant, sans
Oublier tous ses enseignants inoubliables.

Encore on tient à remercier nos familles est nos amis pour leurs encouragements.

« Ne restez pas indéfiniment sur la route qui ne mène qu'à
Des endroits connus, abandonnez parfois les sentiers battus et
entrez dans la forêt, vous découvrirez certainement quelque
Chose que vous n'avez jamais vu, bien sur ce ne sera qu'une
Petite chose, mais prêtez y attention, suivez la, explorer la,
une découverte en amènera une autre, et avant même de vous
Rendre compte, vous aurez mis à jour une idée intéressante »

Alexander Gr ahamBell

Table des matières

Chapitre02: Les motifs séquentiels

Chapitre03: Approches d'extraction de motifs séquentiels

Conclusion générale et perspectives

Liste des figures :

Liste des tableaux :

Introduction générale

Aujourd'hui, le Mégaoctet est l'unité pour la mémoire vive et nous produisons des machines stockant plusieurs Petaoctets (des millions de milliards d'octets). Ce besoin de stockage est justifié et il est indispensable. En effet, depuis quelques années une quantité croissante de données est générée de toute part par des organismes médicaux, industriels, commerciaux, etc... Cet écoulement continue et croissant d'informations peut maintenant être stocké et préparé à l'étude grâce aux nouvelles techniques d'Entrepôt de Données (ou Data Wharehouse). Les fournisseurs de la téléphonie, par exemple, gardent au moins un an les positions géographiques et les consommations de leurs abonnés. Les grands magasins et les entreprises de vente par correspondance (VPC) conservent les achats de leurs clients, collectent des informations sur leurs clients grâce à des systèmes de cartes de fidélité.

Les sites web conservent des traces de connexions sur leurs sites marchands. En résumé les entreprises en secteur très concurrentiel conservent les données de leurs activités et achètent même des données. Les motifs qui ont présidé à la conservation de ces données étaient : des obligations légales pour pouvoir justifier les facturations, des raisons de sécurité pour pouvoir détecter les fraudes, des motifs commerciaux pour suivre l'évolution des clients et des marchés. Quelle que soit la raison initiale, les entreprises se sont rendu compte que ces données pouvaient être une source d'informations à leurs services.

Ce constat, valable pour les sociétés du secteur marchand, peut être étendu à de nombreux domaines et services. Il faut donc définir des environnements permettant de mémoriser de grands jeux de données et d'en extraire de l'information. La complexité et la taille de ces bases de données dépasse la capacité humaine d'analyse. Delà, la nécessité est apparue de développer et appliquer des outils pour extraire des informations utiles. Ces derniers, sont reconnus comme un moyen très efficace d'analyse avancée de données, permettant d'extraire des connaissances cachées depuis de grandes masses de données pour des applications décisionnelles.

Les techniques d'extraction de la connaissance (ECD) communément connus sous le nom de « Data mining » sont utilisées dans le monde professionnel pour l'aide à la décision dans différents domaines d'applications, allant de la gestion de relation client à la maintenance préventive, en passant par la détection des fraudes ou encore l'optimisation dans les sites Web. Citons parmi ces techniques : les arbres de décisions,

l'analyse de liens, la détection de clusters, les algorithmes génétiques, les règles d'association et les règles d'association séquentielles, …etc.

Dans la technique des règles d'association séquentielles, l'extraction des connaissances se fait en recherchant des relations d'ordre, sous forme d'enchaînements appelés séquences entre items (objets, attributs, …) ou ensembles d'items d'une base de données. L'objectif est alors de trouver dans cette base, toutes les séquences d'items apparaissant avec une certaine certitude (selon une mesure d'intérêt choisie par l'utilisateur, par exemple, la fréquence).Il s'agit donc, de construire l'ensemble de tous ces motifs intéressants appelés motifs séquentiels. Ces derniers représentent un outil récent très utile, notamment, aux problématiques relatives à la prédiction de comportements futurs.

Nous proposons dans ce mémoire l'étude de cette technique appliquée à la maintenance des équipements industriels. En effet, les industriels d'aujourd'hui sont de plus en plus équipés de systèmes d'acquisition et de décisions numériques. Ces systèmes, qu'ils agissent au niveau du suivi de la production ou de la maintenance des équipements génèrent des Giga Octets de données. Le volume des données est tel qu'il est désormais impossible à un décideur d'avoir une vue d'ensemble sur ces données. Il est alors nécessaire d'équiper les décideurs dans ce domaine d'outils performants leur permettant d'extraire des informations pertinentes de ces masses de données.

Pour notre choix de la technique des motifs séquentiels, nous mentionnons que ceci a été avant tout inspiré (comme dans la plus grande majorité des applications de Data mining) de la nature du problème que nous avons à résoudre : d'une part, nous nous intéressons à la découverte de modèles d'enchaînement de pannes sur les équipements industriels (séquences de pannes fréquentes) dans l'objectif d'améliorer le processus de la maintenance, et de l'autre, nous exploitons l'aspect temporel des données que nous avons à analyser. A notre plus grande connaissance, c'est la technique la plus adaptée à ce type de problèmes.

Cependant, et pour la tâche d'extraction de ces motifs, nous soulignons qu'il s'agit d'une phase extrêmement importante dans tout un processus, représentant un problème complexe par rapport à celui des motifs fréquents classiques, notamment, lorsqu'il s'agit de le résoudre avec une conjonction de contraintes comme nous le découvrons.

Notre travail consiste dans un premier temps à étudier et à comprendre le fonctionnement des algorithmes d'extraction de motifs séquentiels, dans un deuxième temps, à évaluer et à comparer les performances de ces algorithmes en fonction de différents paramètres.

Ce mémoire est organisé comme suit:

- Le premier chapitre présente les fondamentaux du data Mining.

- Le chapitre II introduit les fondements de base de la technique des motifs séquentiels.

- Le chapitre III étudie de plus près le problème d'extraction de motifs séquentiels dans une base de données temporelle et dresse un état de l'art sur les différentes approches existantes dans ce domaine. Un bilan des méthodes abordées et une étude comparative des algorithmes les implémentant sont effectués sur la base de critères (notamment, de performance) que nous avons défini à cet effet.

- Le chapitre IV présente notre contribution algorithmique dans le domaine de l'extraction des motifs séquentiels. Il introduit notre nouvelle méthode de découverte de ces motifs ainsi que l'algorithme de recherche proposé. Un exemple qui illustre l'algorithme proposé sur une BDD de la maintenance est effectué en fin du chapitre avec la présentation des résultats obtenus.

Et pour finir, nous conclurons tout en proposant des perspectives.

1

Data mining

1. Introduction

L'énorme développement de l'activité humaine ces dix dernières années, autour d'applications informatiques, ne cessant d'être de plus en plus consommatrices de puissances de calcul, de capacités de stockage et de vitesses de transmission des réseaux. De ce fait les systèmes d'information actuels mais également toutes les autres formes de gestion de l'information sont de plus en plus alimentés automatiquement, ce qui a conduit à collecter et accumuler des quantités de données gigantesques ! Certains experts estiment même, que le volume de données numériques conservées au niveau de notre planète, double tous les ans !

Dans cette disponibilité de données de tous types et de toutes origines, encadrant presque tous les domaines d'activités actuelles, une idée très intéressante a vu le jour : il s'agit de pousser l'exploitation de grandes masses de données, au-delà de leur utilisation opérationnelle et fonctionnelle classique connue, par l'insertion de nouveaux outils d'analyse, non statistiques à priori, qui permettrons aux utilisateurs métiers, de faire apparaître à partir de leurs données, des informations qui étaient jusque-là inconnues ou implicites.

Cette démarche actuellement connue sous le nom de "Data mining" ou "fouille de données", permet d'extraire un maximum d'informations, appelées connaissances, cachées dans les données et leur présentation sous forme de règles, de modèles, de régularités ou de concepts. Le processus général d'extraction de la connaissance à partir des données, ne se limite pas à ces deux tâches seulement, il comporte plusieurs étapes pendant lesquelles l'expert humain joue un rôle important, allant de la sélection des données à analyser jusqu'à l'interprétation et la visualisation des résultats, en passant par l'analyse proprement dite.

Les données sources résident souvent dans des bases de données ou dans des entrepôts de données mais également, dans des bases de données distribuées (Distributed Data mining) ou sur Internet (Web Mining). De plus, le Data mining ne se limite pas au traitement de données structurées, il offre des moyens pour aborder les corpus en langage naturel (TextMining), les images (Image Mining), le son (Sound Mining), la vidéo (VideoMining), …etc.

La performance des systèmes informatiques actuels et la maturité des outils d'apprentissage automatique, rendent cette technologie très attrayante à de nombreux domaines d'application : entreprises de la grande distribution, télécommunications, institutions financières (banques, bourses, assurances, …etc.), recherche médicale et scientifique, planification des villes, météorologie, accidentologie, agriculture ou encore le domaine industriel [01], …etc.

Le présent chapitre, présente la démarche par laquelle le Data mining opère pour faire émerger ces différentes applications. Il est organisé comme suit : d'abord, nous introduisons la définition du Data mining, puis nous le situons dans son contexte général, le processus ECD, ensuite expliquons par domaine, quelques-unes de ses importantes applications. Ensuite, nous présentons, les tâches qu'il peut accomplir ainsi que les techniques mises en œuvre tout en associant à chaque tâche les outils les plus appropriés. Enfin, une conclusion sur des points importants abordés clôture ce chapitre.

2. Le Data mining au centre de l'ECD

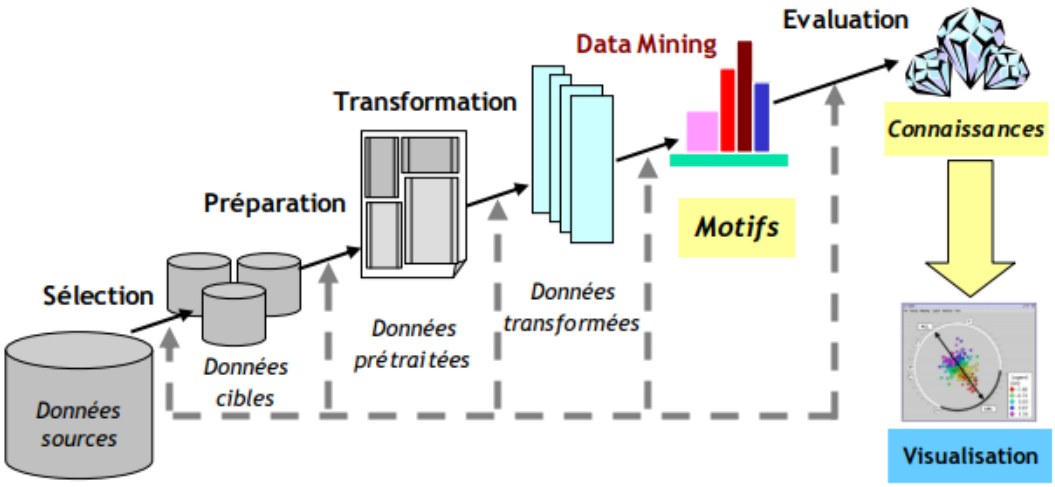


Figure 1.1– Processus ECD d'extraction des connaissances à partir des données

En réalité, le Data mining ne se limite pas à l'extraction automatique des connaissances cachées dans les données et leur présentation à l'utilisateur. Il s'inscrit dans un processus plus général comme élément essentiel. Il s'agit du processus d'Extraction des Connaissances à partir des Données, l'ECD, ou Knowledge Discovery in Databases (KDD).

Le processus ECD peut être vu alors comme la démarche générale de Data mining ou le processus qui transforme les données en connaissances. Cette démarche comporte en effet, plusieurs étapes pendant lesquelles l'expert humain joue un rôle important, allant de la sélection des données à analyser jusqu'à la présentation des résultats à l'utilisateur. C'est un processus itératif et interactif constitué généralement des six étapes suivantes :

(1) Identification du problème et définition des objectifs.

(2) Sélection des données d'analyse.

(3) Nettoyage et enrichissement des données.

(4) Transformation et réduction de la dimension.

(5)Data mining.

(6) Évaluation, validation et visualisation des résultats.

Il est indispensable de mentionner que plusieurs itérations de l'ensemble ou d'une partie de ce processus peuvent être nécessaires afin de répondre à un problème donné (voir Figure03).

2.1. Identification du problème

L'identification du problème et la définition des objectifs représentent en fait, une phase essentielle dans la réalisation de tout processus d'extraction de connaissances à partir des données, qui ne peut en aucun cas, être sous estimée par rapport aux autres phases du processus. Sans cette étape, aucun projet de Data mining ne serait réussi et d'ailleurs, c'est un facteur clé du succès du Data mining. Considérons l'exemple d'une société de télécommunications qui cherche à identifier des paires de clients qui s'appellent souvent de manière à leur proposer par exemple, une offre spéciale, ou une entreprise de e−commerce qui obtient de mauvais taux de réponses aux mailings qu'elle envoie et cherche une manière à mieux cibler son courrier. Sans se fixer ces objectifs au préalable, aucune démarche de Data mining réussie pour ces deux entreprises ne pourra être initiée ou envisagée.

2.2. Sélection des données

La première phase opérationnelle consiste à présélectionner et collecter les données nécessaires au traitement. Il s'agit de faire l'inventaire aussi bien des données a priori utiles pour atteindre les objectifs préalablement fixés, que des données réellement disponibles. Ces dernières, appelées données sources, résident souvent dans des bases de données ou dans des entrepôts de données (Data Warehouse) mais également sur Internet (Web Mining) ou dans des bases de données distribuées (Distributed Data mining). La tendance actuelle est au Data Warehouse. En effet, les entreprises tentent de regrouper les informations dont elles disposent au sein de leurs différents départements en un immense entrepôt de données, qui servira de support d'aide à de multiples décisions. Ce transfert doit se faire régulièrement via des formules sophistiquées, élaborées spécifiquement par des spécialistes. De plus, les données doivent être représentées dans le Data Warehouse, sous une forme actionnable (utilisable), qui permet des opérations de Data mining.

2.3. Nettoyage et enrichissement des données :

Le nettoyage des données intervient immédiatement après leur sélection. Les politiques adoptées pour accomplir cette tâche dépendent généralement du cas particulier considéré. Le traitement des données manquantes relève souvent de leur remplacement par des estimations calculées (moyennes, moyennes conditionnelles, …etc.). De plus, si les données disponibles semblent insuffisantes ou inadéquates pour la tâche prévue, un enrichissement devient nécessaire. Trois types d'enrichissement peuvent être envisagés:

→**Interne :** Retourner vers le système d'information pour essayer d'intégrer une autre source aux données d'analyse.

→ **Externe** : Si les données disponibles en interne semblent inexploitables sans une donnée extérieure identifiée, on peut envisager d'acquérir (importer) cette donnée.

→**Par transformation :** C'est l'enrichissement le plus souvent pratiqué, il s'agit de créer de nouveaux indicateurs par calcul sur les données disponibles.

2.4. Transformation et réduction de la dimension

Les données doivent généralement être transformées, afin de disposer à la fin d'une présentation standard et uniforme, compatible avec la technique de Data mining qui va être appliquée, car elles ne le sont généralement pas (le même format adéquat).

Une fois les données préparées et présentées sous le format standard, elles vont pouvoir être analysées. Cependant, pour les très grands volumes de données, l'analyse nécessite une étape intermédiaire, c'est la réduction de la dimension. En effet, aussi surprenant que cela puisse paraître, les meilleurs modèles sont parfois obtenus grâce à des groupes de données réduits. Les outils de Data mining peuvent potentiellement s'adapter à des données de toute taille, mais, plus un modèle comportera de données, plus de cas exceptionnels il devra intégrer. L'étape de réduction de la dimension élimine ce risque et permet aux programmes d'analyse une étude moins coûteuse en temps de calcul et en espace mémoire.

2.5. Data mining

Maintenant que les données sélectionnées sont traitées et prêtes à l'analyse, il est temps d'appliquer des méthodes intelligentes de Data mining à ces données, afin de déduire des hypothèses et construire des modèles informatiques (patrons, patterns, motifs, …), servant à bien accomplir les tâches spécifiées dans la phase de définition des objectifs.

Cette tâche représente la phase centrale et indispensable de tout le processus. En effet, si elle se limitait à seulement regarder ou explorer les données, des outils d'analyse multidimensionnelle, type OLAP (On−Line Analytical Processing), suffisaient largement. Cependant, et du fait que l'opération se base particulièrement sur la déduction d'hypothèses, les outils de Data mining se révèlent alors incontournables. La construction de modèles ou la modélisation des données par ces outils, peut se faire de deux manières différentes :

(1)Dirigée : Il est possible d'analyser les données de manière dirigée ou supervisée : il s'agit dans ce cas, de résoudre un problème bien déterminé.

(2) Non dirigée : L'analyse est effectuée de manière exploratoire ou non dirigée, également dite non supervisée, sans idée à priori des résultats qui vont en découler.

Dans la première, il s'agit de résoudre un problème spécifique bien déterminé (rencontré ou qui aurait été posé). Pour un problème donné, une hypothèse est formulée à priori. Il s'agit alors de tester celle-ci sur la base des données récoltées dans ce but, par exemple, déterminer les risques cardio-vasculaires que peut avoir un nouveau patient, en se basant sur des données exprimant les problèmes cardio-vasculaires d'un ensemble de patients. Le point de départ est donc l'hypothèse formulée (pour le problème spécifique à résoudre) et toute l'analyse est motivée et dirigée par celle-ci. Ce mode d'analyse est également dit prédictif.

Dans la seconde, il s'agit d'analyser les données en vue d'en dégager une connaissance quelconque, sans aucune hypothèse formulée ou idée à priori des résultats qui vont en découler. Ce mode d'analyse est utilisé lorsque le chercheur de la connaissance ne se doute à priori pas ou très peu de ce qu'il peut découvrir, comme par exemple, créer différents groupes de textes, à partir d'un ensemble de textes de tous genres, selon la similarité de leurs contenus. Dans ce mode d'analyse, également dit déductif, ce sont les données dont on dispose qui dictent les hypothèses à tester et qui amènent à la découverte d'un nouveau savoir.

En somme, les techniques d'analyse non dirigées s'effectuent avant les méthodes d'analyse dirigées quand la quantité de données disponibles est insuffisante, ou que l'on n'a pas d'idée précise sur les informations que celles-ci peuvent contenir. Dans ce cas, elles peuvent se contenter, par exemple, d'identifier des structures au sein des données que l'analyse dirigée tentera d'expliquer. Cet enchaînement d'analyses permet souvent d'offrir un support d'aide à la décision, mais présente parfois, le problème de la pertinence d'un certain nombre de modèles (motifs) extraits qui peut être discutée.

2.6. Evaluation, validation et visualisation des résultats

Cette phase représente l'ultime étape de l'ECD. Elle consiste souvent en les trois tâches suivantes dans l'ordre : l'évaluation, qui mesure la fiabilité des modèles extraits, puis la validation des modèles intéressants par les experts du domaine et enfin, la présentation des résultats à l'utilisateur sous une forme claire et compréhensible.

La fiabilité des modèles extraits est analysée en estimant leurs taux d'erreur car ils ne peuvent être utilisés directement (ou généralisés) en toute fiabilité. Nous devons les évaluer en les soumettant à l'épreuve de la réalité afin d'apprécier leur justesse. Il s'agit d'appliquer le modèle aux données réelles pour voir comment les hypothèses se vérifient. Un modèle qui a été bien entraîné doit normalement avoir un taux d'erreur assez faible lors du traitement de nouvelles données, et ce n'est qu'à partir de ce moment, que l'on peut employer le terme connaissances, à condition que ce modèle soit validé par les experts du domaine.

La phase aval intervient juste après cette étape, il s'agit de la présentation des résultats où les modèles validés sont généralement post-traités (simplifiés, agrégés, transformés, …etc.) afin d'être présentés à l'utilisateur sous une forme intelligible et interactive, permettant une exploitation facile et optimale. L'adoption d'un mode de visualisation et d'un mode opératoire adaptés au type de motifs extraits et aux attentes des utilisateurs, peut être facilitée à l'aide d'outils de visualisation de connaissances (Knowledge

Visualisation Tools). La figure suivante montre deux modes différents de visualisation à l'aide de ces outils.

Exemple

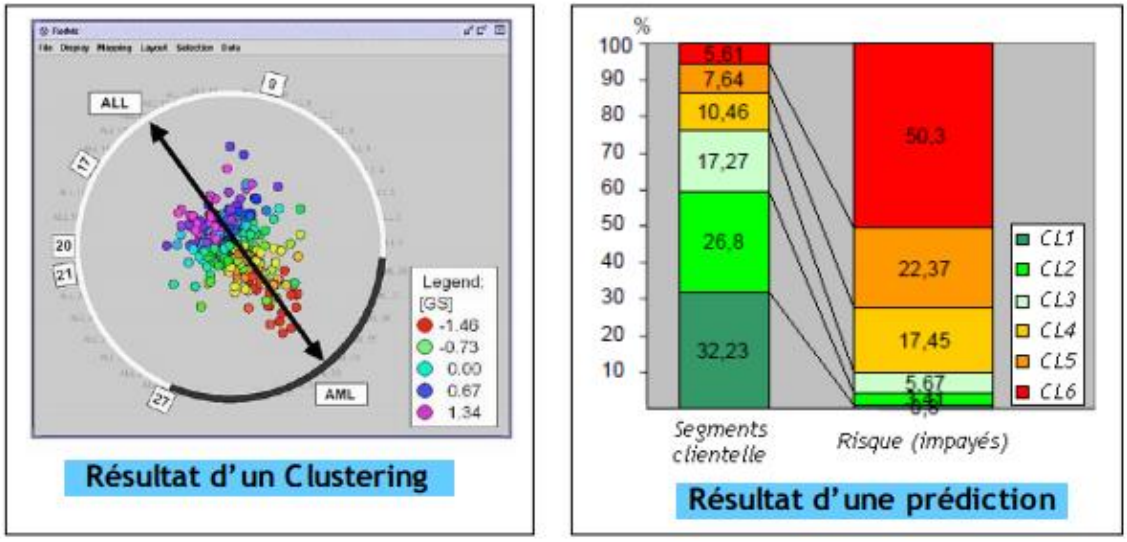


Figure 1.2 – Présentation des résultats à l'aide d'outils de visualisation de connaissances

Les outils de visualisation de connaissances font généralement intervenir des notions d'interaction homme−machine, de convivialité et de visualisation. Nous citons parmi les techniques utilisées, celle des coordonnées parallèles. Du fait de l'importance accordée de plus en plus au rôle de l'utilisateur en Data mining, ces outils sont devenus des composantes majeures dans toute bonne application d'extraction de la connaissance.

Le dernier point important concerne la qualité des modèles qui ne peut être définie seulement à partir de leur fiabilité, car elle est généralement fonction d'une conjonction d'autres critères comme les performances obtenues, la compréhensibilité, la rapidité de construction et d'utilisation et enfin l'évolutivité. Citons parmi les outils de mesure de la fiabilité les matrices de confusion et parmi les outils de mesure des performances les courbes de comparaison (ROC, de Lift et Gain) qui sont parmi les plus utilisées.

3. Le cercle vertueux du Data mining

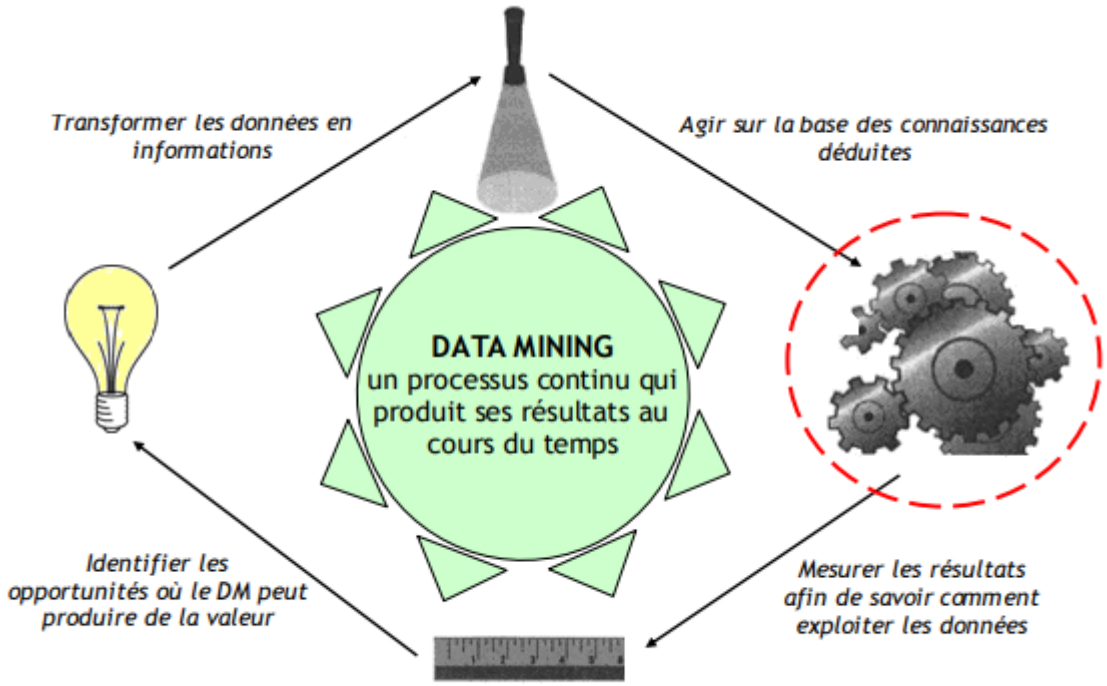


Figure 1.3 – Le cercle vertueux du Data mining

La solution apportée par le Data mining est en réalité plus qu'un simple ensemble de techniques puissantes et de structures de données. Les techniques de Data mining doivent avant tout, être appliquées dans les bons domaines et sur les bonnes données. Le cercle vertueux (Figure 01) rappelle ici qu'il ne s'agit pas d'une étape du processus qui applique les connaissances produites par une meilleure connaissance des clients, des marchés, des produits et de la concurrence à un processus interne. Il s'agit d'un processus continu qui produit ses résultats au cours du temps.

4. Définition du Data mining

L'expression "Data mining" est un néologisme américain, qui signifie "recherche de pépites d'informations utiles dans un grand ensemble de données". On le traduit en français par des expressions comme "fouille de données" [02] ou "forage de données». Comme dans tout forage, son but est de permettre d'extraire un élément : la connaissance. Ses concepts s'appuient sur le constat qu'il existe au sein de chaque entreprise des pépites d'informations utiles enterrées dans le gisement de données. Ils permettent, grâce à un certain nombre de techniques spécifiques, de faire apparaître ces connaissances et leur présentation sous forme de règles, de modèles, de régularités ou de concepts. Dans [03], le Data mining est défini comme le processus d'extraction

d'informations intéressantes non triviales, implicites, préalablement inconnues et potentiellement utiles, à partir de grandes masses de données. Nous appelons Data mining l'ensemble des techniques qui permettent de transformer les données en connaissances.

Data mining et statistique

Les informaticiens ont depuis longtemps fait appel à des méthodes d'analyse, d'apprentissage et de modélisation statistiques telles que les probabilités et les distributions et pourtant beaucoup de scientifiques considèrent toujours, souvent par méconnaissance ou par confusion, le Data mining comme une autre forme d'application de la statistique. La réalité est que le Data mining est une discipline née en dehors de la statistique, dans la communauté des bases de données et de l'intelligence artificielle dans le but de valoriser les grandes bases de données. Le Data mining offre des perspectives nouvelles pour la statistique et répond au défi du traitement des énormes bases de données. Il peut être considéré comme une nouvelle branche enrichissante de la statistique exploratoire, vouée à la découverte de structures inconnues et utiles dans les données.

5. Les types de données qui sont appliqués par la fouille de données

En principe, la fouille de données peut s'appliquer à tous les types de données. Toutefois, selon chaque type de données, les algorithmes de la fouille de données diffèrent. Quelques exemples de types de données (Zaïane (1999)) auxquels peut s'appliquer la fouille de données sont:

"Flat file" les fichiers plats: sont actuellement la source de donnée le plus commune pour les algorithmes du fouille de donnée, ils sont des fichiers en format texte ou binaire, contenant un enregistrement par ligne, avec des champs séparés par des délimiteurs, tels que les virgules ou les tabulations. Dans ce type de fichiers, les données peuvent être des transactions, des séries temporelles etc. .

Base de données relationnelle : est une base de données consistant dans des tableaux séparés, avec des liaisons explicitement dénies et dont les éléments peuvent être combinés sélectivement comme des résultats à des interrogations. Chaque tableau contient des colonnes (correspondantes à des tuples) et des lignes (correspondantes à des attributs), La fouille de donnée peut profiter du SQL pour la sélection, la transformation et la consolidation;

Les data warehouses: est un support de données dans laquelle est centralisé un volume important de données consolidées à partir des différentes sources de données (souvent

hétérogènes), par exemple Si le directeur de l'entreprise veut accéder aux données de tous les magasins pour prendre des décisions stratégiques, il serait plus approprié si toutes les données étaient stockées dans un seul emplacement avec une structure homogène qui permet l'analyse interactive des données. Autrement dit, les données de différents magasins peuvent être chargées, nettoyées, transformées et intégrées ensemble. Pour faciliter la prise de décisions et les vues multidimensionnelles.

Base de données transactionnelle : est un ensemble d'enregistrements représentant des transactions, Une transaction contient un identifient unique (transaction ID) et une liste d'items composant la transaction.

Bases de données orientées objet et relationnelle objet : Il s'agit d'un type spécial de base de données (ou base de données relationnelle) où les données sont des objets.

Les bases de données multimédia : comportent des documents sonores, des vidéos, des images et des médias en textes et audio. Elles peuvent être stockées sur des bases de données orientées objets ou objets relationnelles ou simplement sur un fichier système. Le multimédia est caractérisé par sa haute dimension ce qui rend le datamining sur ce type de données très difficile.[04]

6. Domaines d'application du Data mining

Dans nos jours la fouille de donnée est devenue de plus en plus applicable dans différentes domaines et secteurs d'activités parmi lesquels :

6.1. Vente, distribution / Marketing :

- La gestion de la relation client (GRC ou CRM) consiste en l'ensemble des activités visant à cibler, attirer et conserver les "bons" clients.
- Détection d'associations de comportements d'achat.
- Découverte de caractéristiques de clientèle.
- Prédiction de probabilité de réponse aux campagnes de mailing.

6.2. Evaluation de risques et détection de fraudes

- Crédits bancaires
- Détection de fraudes

6.2. Applications scientifiques

- Diagnostic et recherche médicale
- Identification des thérapies à succès (combinaison de prescriptions).

6.3. Internet

6.4. Analyse de documents texte [07].

7. Les Tâches du Data mining

Contrairement aux idées reçues, le Data mining n'est pas le remède miracle capable de résoudre toutes les difficultés ou besoins d'une entreprise ou d'un organisme quelconque. Cependant, une multitude de problèmes d'ordre intellectuel, économique ou commercial peuvent être regroupés dans leur formalisation, dans l'une des tâches suivantes : [05]

– Classification.

– Estimation.

– Prédiction.

– Groupement selon les affinités.

– Segmentation (Clustering).

– Description.

7.1. Classification

La classification consiste à classer les données (objets, individus, éléments, …) dans des catégories prédéfinies selon les valeurs de leurs caractéristiques. Il s'agit donc d'examiner les caractéristiques d'un objet et de lui attribuer une classe dans un champ particulier à valeurs discrètes. Les classes sont caractérisées à priori et on peut posséder d'un fichier d'apprentissage avec des exemples pré-classés. On construit alors, une fonction qui permettra d'affecter à telle ou telle classe un nouvel objet. La classification est une tâche très utilisée de Data mining.

7.2. Estimation

L'estimation comme son nom l'indique, permet d'estimer des données en se basant sur des valeurs continues à l'opposé de la classification qui traite des valeurs discrètes. En d'autres termes, la classification consiste à estimer la valeur discrète du champ

identifiant de la classe où l'objet serra affecté tandis que dans l'estimation, le champ à estimer est un champ à valeurs continues comme par exemple, la durée de vie d'un client, le pourcentage de clients à risque dans une catégorie, la probabilité de réponse à un mailing, …etc.

Toutefois, l'estimation peut être utilisée dans un but de classification : il suffit d'attribuer une classe particulière par intervalle de valeurs du champ estimé. Par exemple, en estimant les revenus de ménages selon divers critères (type de véhicule et nombre, profession, type d'habitation, catégorie socioprofessionnelle, …etc.), il sera possible de définir des tranches de revenus permettant de les classifier ou de les ordonner pour ne retenir, si on le désire, que les ménages de meilleurs revenus. Cette démarche est souvent pratiquée en marketing pour proposer des offres aux clients potentiels.

7.3. Prédiction

La prédiction est une tâche très attrayante à de nombreuses applications. Cette fonction peut être comparée à la classification ou à l'estimation à la différence que ses observations sont classées ou estimées sur la base de la prévision d'un comportement futur. En d'autres termes, les valeurs connues sont historisées et on cherche à prédire la valeur future d'un champ. Tout comme les tâches précédentes, elle s'appuie sur le passé mais son résultat se situe dans un futur généralement précisé. On comprend facilement, que la seule manière de mesurer la qualité d'une prédiction est d'attendre ! Des exemples d'applications de la prédiction sont :

– Economie : la prédiction de valeurs futures d'actions et d'indicateurs boursiers, la prédiction des prix futurs de matières brutes (pétrole, gaz, or, blé, …etc.).

– Marketing : la prédiction au vu de leurs comportements passés des départs de clients.

– Industrie : la prédiction de pannes futures dans équipements industriels. [06]

– Diagnostic médical : la prédiction de l'efficacité d'un traitement médical.

– Météorologie : les prédictions météorologiques.

7.4. Groupement selon les affinités

Le groupement selon les affinités également dit groupement par similitudes, consiste à grouper les objets qui vont naturellement ensemble. Il est assimilé à la technique des règles d'association. Il s'agit par exemple, de déterminer quels produits sont généralement achetés ensemble lors d'une même transaction, ou quels types de pannes

vont souvent ensemble dans un équipement. On comprend facilement, que la technique la plus appropriée à cette tâche est celle des règles d'association.

7.5. Clustering

Le clustering ou clusterisation permet la segmentation d'une population hétérogène d'objets en un ensemble de sous groupes de populations plus homogènes, appelés clusters ou segments. Contrairement à la classification, qui se base sur des classes prédéfinies ou des exemples pré-classés, les sous groupes en clustering ne sont pas caractérisés à priori : ils sont définis exclusivement en fonction des données d'analyse. Le clustering est une tâche très utilisée de Data mining.

7.6. Description

Parfois le but est simplement de décrire ce qui se passe au sein d'une base de données complexe de manière à augmenter la compréhension de celle-ci. Ceci permet souvent de construire une idée plus ou moins détaillée sur le processus qui est à l'origine de ces données. Il s'agit de manière générale, du démarrage d'une étude où l'on a peu de connaissances sur le phénomène étudié. La description obtenue, permet dans la majorité des analyses de ce type, d'enchaîner sur une ou plusieurs autres tâches du Data mining.

8. Les Outils du Data mining (méthodes)

Maintenant qu'on a décrit les différentes tâches d'analyse rendues possibles grâce au Data mining, il est temps de passer en revue les outils qui permettent de réaliser ces tâches. Seulement, le domaine du Data mining étant tellement vaste, qu'il est impossible de détailler ici chacun de ces outils.

Nous nous contentons alors de leur description dans l'ordre :

– Le raisonnement à base de cas.

– La détection de clusters.

– L'analyse de liens.

– Les arbres de décision.

– Les réseaux neuronaux artificiels.

– Les algorithmes génétiques.

– Les règles d'association.

– Les motifs séquentiels.

Ces outils proviennent en fait de disciplines scientifiques diverses : les algorithmes génétiques et les réseaux neuronaux artificiels ont été inspirés de la biologie, le raisonnement à base de cas appartient au domaine de l'intelligence artificielle et l'analyse de liens découle de la théorie des graphes. Les statistiques marquent également leurs empreintes dans ce domaine. C'est pourquoi nous insistons sur le fait, qu'aucun de ces outils ne peut résoudre seul tous les types de tâches, chaque outil a ses spécificités et usages propres.

8.1. Le raisonnement à base de cas

L'être humain base la plupart de ses décisions sur des expériences passées. Lorsqu'un individu réagit face à une situation quelconque, son processus de décision consiste à faire appel à sa mémoire pour identifier certains cas vécus précédemment comme étant voisins ou similaires à cette situation, et à prendre une décision en fonction de ces voisins.

De façon similaire, la technique de Data mining raisonnement à base de cas RBC (Case BasedReasoning, CBR) également dite raisonnement basé sur la mémoire (Memory BasedReasoning) ou méthode des plus proches voisins PPV (NearestNeighbor), permet de classer ou de prédire des données inconnues à partir d'instances connues. En conservant une base de données de cas passés, cette technique recherche au sein de celle-ci, des cas similaires ou voisins au cas à analyser soit pour les classer lorsque le but est la classification, soit pour en faire une prédiction.

8.2. La détection de Clusters

La détection de clusters ou Clustering, également appelée segmentation, permet de créer des modèles repérant des groupes d'objets similaires. Pour cela, la technique cherche à découvrir des relations intéressantes qui peuvent exister implicitement entre les objets, ce qui permet de regrouper dans un même groupe (Cluster) des objets considérés comme similaires, afin d'en constituer des classes, comme le montre la Figure1.5 Etant donné que les groupes sont inconnus à priori, il s'agit avant tout d'une technique non dirigée de Data mining.

8.3. L'analyse de liens

L’analyse de liens est un outil de Data mining qui cherche à découvrir des relations qui peuvent se tisser entre les différents objets d’une population donnée, par exemple des consommateurs, en vue de dégager des catégories qui pourraient être étudiées séparément. Cette technique est généralement basée sur la théorie des graphes. Un graphe est une représentation intuitive constituée de nœuds et d’arrêtes. Les nœuds symbolisent les objets qui sont en relation et les arrêtes évoquent les relations entre paires d’objets.

8.4. Les arbres de décision

Les arbres de décision sont utilisés afin de répartir une population d'objets en sous groupes homogènes, selon un ensemble de paramètres discriminants définis à priori en fonction de l’objectif de l’analyse. A ce titre, il s’agit avant tout d’une technique dirigée de Data mining. Les arbres de décision représentent des outils puissants et connus, tant pour la classification que pour la prédiction. Ils non seulement, permettent de distinguer les différentes classes et leur associent une ou plusieurs règles, mais présentent l’atout de représenter des règles très facilement interprétables, comme le montre l’exemple suivant.

8.5. Les réseaux neuronaux artificiels

Avec le développement des puissances de calcul, les réseaux neuronaux artificiels sont devenus des outils parmi les plus répandus et évolués du Data mining. C'est une technique issue de la biologie. En effet, un réseau de neurones artificiel est un modèle de calcul dont la conception est très schématiquement inspirée des interconnections du cerveau humain, ce qui permet au réseau de prendre des décisions s’appuyant davantage sur la perception que sur le raisonnement logique formel. De plus, les modèles dont ils se basent sont généralement optimisés par des méthodes d’apprentissage statistique, telles que les probabilités et les distributions, leur permettant de générer de vastes espaces fonctionnels souples.

Un réseau de neurones artificiel possède généralement d’un algorithme d’apprentissage, qui consiste à auto-modifier des paramètres réels en entrée du réseau, appelés coefficients ou poids synaptiques en fonction d’un jeu de données exemple présenté en entrée. Le but de cet entraînement est de permettre au réseau, d’apprendre à partir d’exemples d’apprentissage. Si l’entraînement est correctement réalisé, le réseau est capable de généraliser à partir de ce jeu, ce qui représente tout l’intérêt des réseaux de neuronaux artificiels.

8.6. Les algorithmes génétiques

Comme leur nom l'indique, les algorithmes génétiques représentent une technique dont le principe est inspiré des mécanismes génétiques naturels. En effet, afin d'élaborer des solutions parmi plus optimales, cette technique crée plusieurs générations de solutions qui évoluent au cours du temps afin de toujours obtenir la solution la plus performante.

L'évolution se fait suivant trois procédés : la sélection, le croisement et la mutation. Le processus de sélection est similaire à celui de la sélection naturelle : uniquement les meilleures solutions se reproduisent. Le croisement se produit quand deux individus choisis au hasard parmi la population sont reliés (accouplés) de sorte que le résultat contient une partie de l'information de chacun de ses parents. Enfin, la mutation arrive soudainement lorsqu'il y a une erreur dans la transmission du code génétique des parents à l'enfant.

8.7. Les règles d'association

La méthode des règles d'association ou l'analyse du panier de la ménagère (Market Basket Analysis) est une technique de Data mining qui a vu le jour dans le domaine de la grande distribution. Le principe consistait à déterminer des combinaisons de produits (appelés items) que le client moyen achetait souvent simultanément. Le modèle qui en découle est très simple : il donne la probabilité que plusieurs produits déterminés soient achetés dans la même transaction. Ces probabilités sont exprimées sous forme de règles. Une règle d'association s'écrit généralement de la forme : {ensemble1 d'items} $\rightarrow$ {ensemble2 d'items}.

Considérons l'application typique de la gestion de tickets de caisse d'un supermarché. On sera par exemple capables, à l'aide d'une règle d'association extraite à partir de la base de données correspondante de dire que, "60% des clients qui achètent le produit Café ont également tendance à acheter le produit Sucre dans la même transaction". Ce qui peut être écrit sous la forme : "Café $\rightarrow$ Sucre : 60%". Une telle constatation est très intéressante puisqu'elle peut aider le gestionnaire à maximiser ses ventes et optimiser ses achats.

Cette technique peut être appliquée à **tout domaine** pour lequel il est intéressant de rechercher des groupements potentiels de produits ou de services, comme par exemple : les services bancaires (identification de types de services très demandés conjointement), la recherche médicale (identification de complications dues à des associations de médicaments), les télécommunications (détection de combinaisons de

facteurs agissant sur les performances des réseaux), la maintenance (détection des pannes qui vont fréquemment ensemble), la détection de fraudes (détection d'associations inhabituelles), …etc.

8.8. Les motifs séquentiels

Contrairement aux règles d'association qui identifient les items qui occurrent simultanément dans une base de données, les motifs séquentiels ou règles d'association séquentielles permettent d'analyser l'ordre d'apparition des items dans cette base, selon un identifiant temporel (par exemple, la date). Ainsi, un motif séquentiel permet par exemple, d'identifier une relation de la forme : "30% des clients qui achètent du Café, achètent également du Sucre dans les cinq jours qui suivent". Ce type de connaissances n'était pas envisageable de formuler ou de découvrir avec les règles d'association classiques.

Autrement dit, les motifs séquentiels tentent d'identifier des enchaînements particuliers d'items dans une base de données. Par conséquent, l'ordre d'apparition des items est fondamental dans ces règles [07]. Considérons l'exemple de la règle classique "Café $\rightarrow$ Sucre", elle n'est pas équivalente au motif séquentiel noté "" qui signifie : "Sucre après Café", puisque dans la première les achats de Café et Sucre se sont survenus dans la même transaction, alors que dans le second, l'achat du produit Sucre est survenu après l'achat du produit Café. C'est pourquoi, les motifs séquentiels tirent leur nom du fait que la séquence joue un rôle majeur dans ce type de règle.

Grâce à la prise en compte de la temporalité, les motifs séquentiels présentent des règles plus précises et plus utiles pour une diversité d'applications actuelles, allant de la prédiction des chemins de navigation des utilisateurs Web et la détection d'enchaînements de transactions financières inhabituelles jusqu'à l'analyse des séquences ADN en passant par la maintenance industrielle anticipée. La recherche de motifs séquentiels est une technique **non dirigée** de Data mining car on ne dispose en entrée que des données d'analyse.

9. Conclusion

Les données informatiques sont de valeur inégalable si elles peuvent être exploitées à des fins allant au-delà de leur utilisation fonctionnelle classique. Le besoin et la créativité ont alors donné naissance au Data mining : l'extraction des connaissances à partir des données. Il s'agit de rechercher le maximum d'informations utiles cachées dans les données par des méthodes automatiques et semi-automatiques et l'utilisation opérationnelle de ce savoir. Le Data mining propose d'utiliser un ensemble d'outils

issus de disciplines scientifiques diverses pour construire des modèles exprimant de nouvelles connaissances à l'utilisateur.

Les grands secteurs de l'activité humaine ont tous compris et mis en œuvre le Data mining pour améliorer leurs connaissances afin d'augmenter leur efficacité et leur créativité pour enfin augmenter leur rentabilité. Nous concluons sur ce point que le Data mining peut être appliquée à tout domaine pour lequel le système d'information entretient les données nécessaires et suffisantes pour atteindre l'objectif bien considéré.

Grâce à Internet, une grande quantité de sites regroupant des logiciels, des expertises, des cours, des communautés d'échange et de la bibliographie relative sont à présent accessibles. En nous référant à ces différentes sources et à d'autres parmi les plus intéressantes, nous avons fait à travers ce chapitre un tour d'horizon sur les concepts de base du Data mining. Nous avons détaillé plusieurs de ces applications, notamment, celles que nous avons trouvées moins abordées dans la littérature. Nous avons également situé le Data mining dans le processus général d'extraction des connaissances à partir des données : l'ECD en expliquant ses différentes phases, en particulier, la phase de visualisation des résultats. Ensuite, nous avons décrit les différentes tâches d'analyse que peut accomplir ainsi que les outils permettant de les réaliser. Nous concluons ici qu'aucun des outils de Data mining ne peut résoudre seul tous les types de tâches : chaque outil a ses spécificités et usages propres.

Parmi les outils présentés, la technique des motifs séquentiels, le chapitre 2 prendra en détail cette technique en commençant par les concepts de base des règles d'association car elles en sont à l'origine.

02

Motifs séquentiels.

1. Introduction

La découverte de règles d'association est une technique qui a vu le jour avec la recherche en bases de données et Data mining pour trouver des relations intéressantes entre articles achetés ensemble. Considérons l'exemple de l'application relative à la gestion de tickets de caisse d'un supermarché. On sera par exemple capables, à l'aide d'une règle d'association extraite à partir de la base de données correspondante de dire que, "60% des clients qui achètent du Café ont tendance à acheter du Sucre dans la même transaction", ce qui peut être écrit sous la forme : "Café $\rightarrow$ Sucre : 60%". Une telle constatation est très intéressante puisqu'elle peut aider le gestionnaire à maximiser ses ventes et optimiser ses achats.

Nous comprenons facilement qu'une règle d'association peut être vue comme une relation d'occurrence simultanée forte qui lie deux ou plusieurs items particuliers. Toutefois, l'exploitation de ce type de règles ne tient pas compte de l'ordre dans le temps entre les items. En effet, reprenons l'exemple de la règle Café $\rightarrow$ Sucre, elle indique seulement que les clients qui achètent du Café ont également tendance à inclure du Sucre dans le même achat, mais, ne peut définir un intervalle de temps sur lequel l'article Sucre pourra être acheté. Ce type de règles ne considère donc que les items qui vont ensemble dans la même transaction.

Lorsque l'occurrence des items est étalée selon un ordre spécifique dans le temps, comme l'exemple de la règle : "30% des clients qui achètent du Café, achètent également du Sucre dans les 7 jours qui suivent", les relations entre ces items sont qualifiées de règles d'association séquentielles ou motifs séquentiels. Les motifs séquentiels permettent de définir des règles plus précises et donc plus utiles et trouvent à l'heure actuelle de larges champs d'application allant de l'analyse des séquences ADN jusqu'à la prédiction des comportements en passant par la maintenance préventive (anticipée) des équipements industriels.

Le présent chapitre introduit et explique les fondements de base de cette technique. Il est organisé en quatre sections principales :

– La Section 2 propose de dresser un rappel sur les règles d'association classiques car elles en sont l'origine.

– La Section 3 sera consacrée à la compréhension de la problématique des motifs séquentiels.

–La Section 4 décrit le processus général d'extraction des connaissances ECD adapté à cette problématique. Nous l'étalons notamment sur la phase principale de ce processus, à savoir, l'extraction de l'ensemble des séquences fréquentes.

– La Section 5 propose de détailler la tâche d'extraction de motifs séquentiels dans une base de données avant de présenter l'algorithme général permettant de réaliser cette tâche.

Nous concluons ce chapitre par un résumé des points abordés et une discussion sur les critères de performance dans la recherche de motifs séquentiels dans une base de données.

2. Aperçu sur règles d'association

La technique des règles d'association est une méthode de Data mining qui a fait son début dans le domaine de la grande distribution. Elle consistait à déterminer des combinaisons d'articles que le consommateur moyen achetait lors d'une même transaction. Le modèle qui en découle est très simple : il donne la probabilité que plusieurs articles déterminés soient achetés ensemble. Ces probabilités sont exprimées sous forme de règles.

Considérons l'exemple de la règle d'association extraite d'une base de données relative aux ventes d'un supermarché : Café $\rightarrow$ Sucre [support =7%, confiance=60%]. Cette règle signifie que les clients qui achètent l'article Café ont également tendance à inclure l'article Sucre dans ce même achat. La mesure support définit la proportion de clients qui ont acheté les deux articles Café et Sucre, c'est-à-dire, la portée de la règle, tandis que la mesure confiance définit la proportion de clients qui ont acheté l'article Sucre parmi ceux qui ont acheté l'article Café, autrement dit, la précision de la règle. L'extraction des règles d'association consiste donc à découvrir dans la base de données, toutes les règles dont le support et la confiance sont supérieurs ou égaux, à des seuils minimaux définis par l'utilisateur.

Ces exemples montrent le grand avantage des règles d'association : leur pouvoir explicatif. En effet, et contrairement à d'autres méthodes d'analyse, elles fournissent des réponses simples et sont facilement interprétables quel que soit le degré de complexité de la règle. C'est pourquoi, elles représentent un sujet de recherche très attractif. L'objectif principal de cette technique est descriptif, toutefois, et dans la mesure où les résultats peuvent être utilisés dans le futur, il pourra être considéré comme prédictif. La

recherche de règles d'association est une technique non dirigée car on ne dispose en entrée que des données d'analyse.

Exemples de règles

- {Achat "Télévision"}⟶ {Achat "Décodeur"} [69%].
- {(Traitement X) ∧ (Traitement Y)} ⟶ {Effet E} [50%].
- {(Maladie M) ∧ (Traitement X)} ⟶ {Guérison} [97%].
- {(Défaut A) ∧ (Défaut B)} ⟶ {Panne P} [65%].
- {Panne P}⟶ {(Panne P1) ∧ (Panne P2) ∧ (Panne P3)} [90%]

2.1.Application à différents domaines

Les règles d'association peuvent être appliquées à tout domaine pour lequel il est intéressant de rechercher des groupements potentiels de produits ou de services, comme par exemple, dans le domaine médical, pour l'identification de complications dues à des associations de médicaments ou dans la détection de fraudes par la recherche d'associations inhabituelles. Nous présentons dans ce qui suit, quelques applications intéressantes dont les résultats ont pu être améliorés grâce l'analyse des règles d'association extraites.

- **Planification commerciale :** Les règles d'association apportent une aide importante dans le placement des articles achetés fréquemment ensemble. Les entreprises de vente par correspondance se servent de ces règles afin de déterminer l'organisation de leurs catalogues, définir des catalogues personnalisés en se basant sur les achats précédents des clients, choisir un article à mettre en promotion, cibler les clients potentiels, …etc.

- **Maintenance industrielle :** L'utilisation des règles d'association dans la gestion, le suivi, la surveillance et la maintenance des unités de production permet de mieux organiser les ateliers, mieux contrôler la qualité des produits (en anticipant les défauts), diagnostiquer les anomalies et anticiper les pannes dans les équipements industriels[08]

- **Réseaux de télécommunication :** L'analyse des données relatives aux systèmes et réseaux de télécommunication permet souvent d'identifier des combinaisons de facteurs particuliers agissant sur les performances des réseaux. Des milliers d'alarmes sont détectées sur ces réseaux chaque jour, les règles d'association sont utilisées pour filtrer les alarmes non informatives, analyser les erreurs de transmission dans le réseau, identifier les causes d'anomalies et dans la prédiction de situations alarmantes.

•**Recherche médicale :** Les règles d'association tirées des bases de données des organismes médicaux constituent un vrai outil d'aide au diagnostic médical et à l'identification de traitements appropriés. Elles servent à la découverte de complications dues à des associations de médicaments, l'identification de populations à risque vis-à-vis de certaines maladies et à la prédiction de résultats d'analyses par combinaison de caractéristiques des patients aux résultats d'autres analyses.

•**Internet :** Une quantité énorme d'informations est accessible par le réseau internet et un nombre important d'accès à ces données est réalisé chaque jour. Les règles d'association extraites à partir des historiques de navigation des utilisateurs Web peuvent être utilisées afin de personnaliser la consultation, améliorer les modes d'accès aux informations, réorganiser et optimiser les sites Web, …etc. Des sites d'e−commerce comme Amazon.fr, utilisent ces règles pour faire des suggestions à leurs clients.

•**Applications spatiales** : Les règles d'association sont très utilisées dans l'analyse de bases de données spatiales, notamment celles regroupant des données géographiques, géologiques ou de génie civil, dans le but de définir des relations entre les caractéristiques des objets (forme, dimension, couleur, poids, …etc.) et résoudre des problématiques relatives à ce type d'applications. Elles sont par exemple utilisées dans GeoMiner pour la prédiction d'évènements naturels (éruptions, tremblements de terre, ouragans, …etc.)

2.2. Concepts de base

Une règle d'association peut être vue comme une relation d'occurrence simultanée qui lie deux ou plusieurs items particuliers. Formellement, le problème des règles d'association peut être présenté de la manière suivante (nous gardons la formalisation introduite dans [09]) :

Définition 01 (Item)

Un item est tout article, élément, attribut ou littéral appartenant à un ensemble fini d'éléments distincts $I = \{i1, i2, \ldots, im\}$.

Exemple

Dans une application de type Market Basket Analysis (analyse du panier de la ménagère), les articles en vente dans un supermarché sont des items. L'ensemble I peut contenir les items A, B et C correspondant respectivement aux articles : Café, Sucre et Lait.

Définition 02 (Itemset)

On appelle itemset ou motif tout sous ensemble d'items de l'ensemble I. Un itemset constitué de k items est un k−itemset.

Exemple

L'itemset {A, B,C} est un 3−itemset représentant les articles : Café, Sucre et Lait. Ces trois articles ont pu ou non être achetés ensemble lors d'une même transaction.

Définition 03 (Transaction)

Une transaction est un couple (TID, Itemset) formé d'un identificateur unique de transaction (Transaction IDentifier) et d'un itemset représentant l'ensemble des items de la transaction.

Définition 04 (Base de transactions)

Une base de transactions D peut être vue comme un ensemble de transactions (TID, Itemset).D= {T: TID, Itemset*)/Itemset={a∈I}*}

Exemple

Le Tableau 2.1 représente une base de transactions constituée de six transactions. Chaque transaction peut être vue comme un itemset identifié par le numéro de la transaction, regroupant l'ensemble des items de cette transaction.

TID	*ITEMSET*
1	{A,C,D}
2	{B,C,E}
3	{A,B,C,E}
4	{B,E}
5	{A,B,C,E}
6	{B,C,E}

Table 2.1 – Base de données à six transactions [03]

Définition 05 (Itemset fréquent)

L'itemset X est dit fréquent dans une base de transactions D si son support est au moins égal au seuil de support minimum MinSup défini par l'utilisateur.

X fréquent⇔Sup(X) ≥MinSup

Exemple

Considérons la base de transactions représentée par le Tableau 2.1. Pour un seuil de support minimum, MinSup = 2, on aura l'itemset {A, C,E} fréquent dans cette base. En effet, son support Sup ({A, C, E}) = 2 ≥ MinSup. Ceci n'est pas le cas pour {C, D} qui est de support 1.

Définition 06 (Règle d'association)

Une règle d'association est une implication de la forme : X → Y [Support, Confiance], où X et Y sont deux itemsets tel que X ∩ Y =∅, avec :

− Support

Le support d'une règle d'association X → Y dans une base de transactions D est donné par le nombre de transactions de D contenant tous les items de l'itemset XU Y

$$Sup(X \rightarrow Y) = Sup(X \cup Y)$$

− Confiance

La confiance d'une règle d'association X → Y dans une base de transactions D est donnée par le rapport du nombre de transactions de D contenant tous les items de X U Y au nombre de transactions contenant au moins X.

$$Conf(X \rightarrow Y) = \frac{Sup(X \cup Y)}{Sup(X)}$$

Exemple

Reprenons la base de données à six transactions du Tableau 1 et considérons la règle d'association Café → Sucre.

– Son support est donné par la proportion dans cette base, de clients qui ont acheté les deux articles Café et Sucre simultanément (c'est-à-dire, dans la même transaction), ce qui est égal à 2/6. La mesure de support définit donc, la portée de la règle.

– Sa confiance est donnée par la proportion dans cette base, de clients qui ont acheté l'article Sucre parmi ceux qui ont acheté l'article Café, ce qui est égal à 1/15. La mesure de confiance définit donc, la précision de la règle.

Définition 07 (Règle d'association valide)

Une règle d'association est dite valide ou intéressante dans une base de transactions D si son support dans D est au moins égal au seuil de support minimum MinSup et sa confiance dans D est au moins égale au seuil minimum de confiance MinConf, définis par l'utilisateur.

$$X \to Y \text{ règle valide } \Leftrightarrow \; Sup(X \to Y) \geq MinSup \wedge Conf(X \to Y) \geq MinConf$$

La Définition 07 est fondamentale dans la problématique des règles d'association. En effet, elle définit le problème à résoudre par toute approche visant la découverte de ces règles dans une base de données D : extraire à partir du jeu de données D, l'ensemble de toutes les règles d'association valides ou intéressantes, autrement dit, l'ensemble :

$$R_A = \{ r : l_1 \to (l_2 - l_1) \mid l_1, l_2 \in F \wedge Conf(r) \geq MinConf \} \quad \text{avec}$$

$$Conf(r) = \frac{Sup(l_1 \cup l_2)}{Sup(l_1)} \quad \text{et} \quad F = \{ l \subseteq I \mid l \neq \phi \wedge Sup(l) \geq MinSup \}$$

2.3. Recherche de règles d'association

Le problème de découverte des règles d'association est généralement décomposé en deux sous problèmes, représentant chacun une phase de traitement particulière :

(1) La phase de recherche et extraction de l'ensemble des itemsets fréquents.

(2) La phase de génération des règles d'association à partir de cet ensemble.

2.3.1. Extraction des itemsets fréquents

Il s'agit de trouver l'ensemble F de tous les itemsets fréquents, c'est-à-dire, ceux qui apparaissent dans la base de données avec un support supérieur ou égal à MinSup. La recherche des itemsets fréquents dans une base de données quelconque est un problème non trivial car le nombre d'itemsets potentiellement fréquents (appelés itemsets candidats) est fonction exponentielle du nombre d'items dans cette base ($2^n - 1$). De plus, de nombreux balayages coûteux en temps UC doivent être répétés sur cette base de données afin d'évaluer les supports des itemsets candidats à chaque étape de la recherche.

2.3.2. Génération des règles d'association

Il s'agit de générer l'ensemble RA des règles d'association intéressantes à partir de l'ensemble des itemsets fréquents F trouvé lors de la phase (1). Les règles générées doivent donc avoir une mesure de confiance supérieure ou égale au seuil de confiance

minimum défini par l'utilisateur, MinConf. Généralement, la génération des règles d'association est réalisée de manière directe sans accéder à la base de données, et par conséquent, le coût en temps d'exécution de cette phase est très faible comparé au coût d'extraction des itemsets fréquents.

C'est pourquoi, une grande part des travaux liés à la problématique des règles d'association s'intéresse au premier sous problème. Ainsi, de longs efforts ont été effectués dans le sens de minimiser : l'espace de recherche, le nombre d'itemsets candidats, l'utilisation de l'espace mémoire et nombre de parcours coûteux de la base de données. En somme, les algorithmes de recherche de règles d'association doivent faire face aux défis suivants : [10]

→Trouver de bonnes heuristiques pour élaguer l'espace de recherche dans le but d'ignorer à l'avance et durant toutes les étapes de la recherche tous les itemsets dont le support ne peut être supérieur ou égal au seuil de support minimum.

→Trouver des solutions techniques, notamment des structures de données adéquates, permettant de minimiser les coûts UC et MC des étapes de génération et de calcul des supports des itemsets candidats considérés.

→Trouver les bonnes techniques pour réduire les coûts des opérations d'entrée/sortie, puisque les grandes bases de données à analyser (ce qui est souvent le cas) résident originellement sur une mémoire secondaire.

2.4. Algorithmes d'extraction de règles d'association

Les temps de réponse des algorithmes d'extraction des règles d'association dépendent essentiellement des temps d'extraction des itemsets fréquents. Plusieurs balayages de la base de données doivent être réalisés en comptant pour chaque itemset candidat le nombre de transactions de la base dans lesquelles il est contenu. Le nombre d'itemsets candidats et la taille des jeux de données étant généralement importants, de nombreuses approches visant à minimiser le nombre d'itemsets candidats et le nombre de ces balayages sont proposées.

2.4.1. Algorithmes d'extraction des itemsets fréquents (F)

Les algorithmes d'extraction des itemsets fréquents par niveau considèrent un ensemble d'itemsets d'une taille donnée lors de chaque itération. Ces algorithmes se basent essentiellement sur la propriété d'antimonotonicité du support afin de minimiser le nombre des itemsets candidats générés à partir des itemsets fréquents de l'itération précédente : "Tous les sur-ensembles d'un itemset non fréquent ne peuvent être

fréquents". Citons l'algorithme pionnier de cette approche : APRIORI. Ce dernier réalise un nombre de balayages de la base de données ne dépassant pas la taille du plus long itemset fréquent trouvé. Une variante de celui-ci appelée DIC permet de réduire le nombre de ces balayages en considérant des itemsets de plusieurs tailles lors d'une même itération.

2.4.2. Algorithmes d'extraction des itemsets fréquents maximaux (FM)

Une alternative a été proposée afin de n'extraire que les itemsets les plus longs parmi tous les itemsets fréquents et de déduire le reste des itemsets fréquents à partir de cet ensemble. Il s'agit de l'extraction des itemsets fréquents maximaux (FM), qui sont des itemsets fréquents dont tous leurs sur-ensembles ne le sont pas. L'extraction est réalisée par une exploration itérative de la représentation en treillis des itemsets, en avançant d'un niveau du bas vers le haut et de un ou plusieurs niveaux du haut vers le bas lors de chaque itération.

Une alternative a été proposée afin de n'extraire que les itemsets les plus longs parmi tous les itemsets fréquents et de déduire le reste des itemsets fréquents à partir de cet ensemble. Il s'agit de l'extraction des itemsets fréquents maximaux (FM), qui sont des itemsets fréquents dont tous leurs sur-ensembles ne le sont pas. L'extraction est réalisée par une exploration itérative de la représentation en treillis des itemsets, en avançant d'un niveau du bas vers le haut et de un ou plusieurs niveaux du haut vers le bas lors de chaque itération.

Les algorithmes de ce type permettent de réduire l'espace de recherche et à fortiori, le nombre d'itemsets candidats considérés. Ils permettent également de réduire le nombre de parcours de la base de données. Cependant, ils présentent l'inconvénient de perte d'informations sur les supports des itemsets, indispensables au calcul des mesures de confiance des règles d'association ! L'algorithme pionnier de cette approche est MAX−MINER.

2.4.3. Algorithmes d'extraction des itemsets fréquents fermés (FF)

L'idée consiste à n'extraire qu'un sous-ensemble d'itemsets fréquents fermés (FF) qui constitue un ensemble générateur non redondant minimal pour tous les itemsets fréquents. C'est une représentation condensée qui permet de déduire le support de n'importe quel itemset fréquent sans recourir au parcours de la base de données. Intuitivement, un itemset est dit fermé s'il n'a aucun sur-ensemble avec le même support. L'algorithme CLOSE est considéré comme l'algorithme pionnier de cette approche. [11]

Exemple

Considérons la base de données suivante, à deux transactions seulement :

TID	*ITEMSET*
1	$\{I_1, I_2, \ldots I_{100}\}$
2	$\{I_1, I_2, \ldots I_{50}\}$

Pour un support minimum MinSup = 1, le processus classique génère pas moins de (2100 –1) soit près de 1030 itemsets fréquents ! Cependant, l'approche d'extraction des itemsets fréquents fermés ne génère que les deux itemsets fréquents fermés : {I1, I2,…I50} avec un support de 2 et {I51, I52,…I100} avec un support de 1. Le reste des itemsets fréquents et leurs supports sont dérivés à partir de ces deux seulement, comme par exemple : {I1 , I2}: 2, {I1, I50}: 2, {I52, I53, I54}: 1, {I50, I99, I100}: 1, {I1, I51, I52}: 1, …etc.

2.5. Résumons

Les règles d'association présentent l'avantage d'être facilement interprétables. En effet, elles montrent aisément comment des objets (ou des évènements) se situent les uns par rapport aux autres. Elles trouvent actuellement de larges champs d'application à tout domaine pour lequel il est intéressant de rechercher des groupements potentiels de produits ou de services

Nous avons décrit cette technique tout en précisant les deux phases indispensables à sa mise en œuvre : la phase d'extraction de l'ensemble des itemsets fréquents et la phase de génération des règles d'association à partir de cet ensemble. Nous concluons ici que c'est la première phase qui est la plus coûteuse de tous le processus, car pour la deuxième, les temps de calcul sont faibles puisque aucun balayage de la base n'est nécessaire pour la génération de ces règles. C'est pourquoi, le problème de recherche de règles d'association se restreint généralement au premier sous problème : l'extraction des itemsets fréquents.

Nous avons décrit cette étape puis passé en revue les trois différentes approches relatives à sa résolution. Nous concluons sur chacune de ces approches :

L'approche d'extraction de tous les itemsets fréquents et son principal inconvénient qui est le nombre important de parcours de la base de données en plus de la génération d'un nombre exorbitant d'itemsets candidats pendant le processus de recherche.

L'approche d'extraction des itemsets fréquents maximaux qui propose de minimiser ces deux facteurs mais présente l'inconvénient de perte de l'information sur les supports des itemsets fréquents trouvés.

L'approche conceptuelle basée sur la propriété de fermeture d'un itemset qui consiste à n'extraire qu'un sous-ensemble générique d'itemsets fréquents fermés, présentant l'avantage d'être complet du point de vue de la connaissance tout en étant réduit du point de vue de la taille. Cette approche est particulièrement efficace pour les bases de données denses qui représentent une part importante des bases de données existantes.

Nous proposons dans ce qui suit d'étendre la problématique d'extraction des règles d'association à la considération du facteur temporel dans ces règles. C'est une généralisation très intéressante puisqu'elle permet d'analyser, comme nous le verrons, l'ordre d'apparition des items dans une base de données. Le but étant de permettre d'extraire des enchaînements particuliers d'items ou de groupes d'items intéressant d'autres types d'applications, c'est la découverte des règles d'associations séquentielles ou ″motifs séquentiels″. Nous nous intéressons plus particulièrement dans ce mémoire à ce type de motifs, très adaptés comme nous les découvrons, à beaucoup d'applications.

3.motifs séquentiels

La problématique de l'extraction de motifs séquentiels peut être perçue comme une extension de celle de l'extraction de règles d'association. En effet la prise en compte de la temporalité dans les enregistrements à étudier permet une plus grande précision dans les résultats, mais implique aussi un plus grand nombre de calculs et de contraintes elles représentent une e extension des règles d'association à deux niveaux.

Le premier : Les BDD manipulées par les motifs séquentiels présentent la particularité de comporter pour chaque transaction, en plus de l'identifiant unique TID de la transaction et de l'ensemble des item s de cette transaction, un **Identifiant Temporel** Cet identifiant permet à la base de données d'enregistrer le temps valide pour chaque nouvelle transaction Insérée. Par exemple, dans une base de données relative aux opérations de vente d'un supermarché chaque transaction inclut un attribut supplémentaire qui indique quand la transaction s'est produite (par exemple, le mois, la date, l'heure, …etc.).

Le second : Les motifs séquentiels permettent d'analyser l'ordre d'apparition d'items ou de groupes d'items dans une base de données selon cet identifiant temporel. Ainsi,

une règle d'association séquentielle permet par exemple, d'identifier une relation de La forme : 80% des clients achètent du Sucre après avoir acheté du Café".

Ce type de connaissances n'était pas envisageable sans la considération de cette nouvelle composante temporelle, absente dans la technique des règles d'association classiques.

3.1 .Extraction de motifs séquentiels versus extraction d'itemsets de règles d'association

Contrairement aux règles d'association classiques qui identifient les items qui occurrent simultanément et cherchent des relations entre ces items à l'intérieur de la même transaction, les règles d'association séquentielles cherchent des relations entre items répartis sur l'ensemble des transactions de la base. C'est pourquoi, elles sont qualifiées de règles inter−transactionnelles à l'inverse des règles classiques intra−transactionnelles. Les règles d'association séquentielles tentent d'identifier des enchaînements (souvent dans le temps) d'items ou de groupes d'items et par conséquent l'ordre d'apparition des items par rapport à l'identifiant temporel est un élément fondamental dans ces règles. Considérons l'exemple de la règle d'association classique "Café $\rightarrow$ Sucre". Elle n'est pas équivalente à l'association séquentielle notée "" qui signifie : "Sucre après Café", puisque dans la première, les achats de Café et Sucre se sont survenus dans la même transaction, alors que dans la seconde, l'achat de Sucre est survenu après l'achat de Café, avec un décalage de transaction entre ces deux articles basé sur l'attribut temporel. C'est pourquoi, les motifs séquentiels tirent leur nom du fait que la séquence joue un rôle fondamental dans ce type de motifs. Nous utilisons dans ce qui suit l expression motifs séquentiels plutôt que règles séquentielles pour désigner ces règles.

Exemples de séquences

- < {Achat "Ordinateur"} {Achat "Imprimante", dans 3 mois} > [70%].
- <{ (Traitement X) $\wedge$ (Traitement Y)} {effet E, dans 15 jours} > [50%].
- < {Défaut A}{Panne P1, dans 15 jours} {Panne P2, dans 15 jours} > [98%].
- < {Prêt "Livre 2"} {Prêt "Livre 5"} {Prêt "Livre 1"} > [20%].
- < {"Google.dz"} {"Recherche Algérie"} {"Algérie−monde.com"} > [30%].

3.2. Champs d'application plus étendus

La prise en compte de la temporalité et de l'ordre des items dans les motifs séquentiels rend ces derniers très utiles et attractifs pour une diversité d'applications actuelles, allant de la planification commerciale jusqu'aux applications spatiales en passant par les applications sur les réseaux de télécommunication et la maintenance industrielle sans compter l'analyse des historiques des accès Web et l'aide au diagnostic médical. [12] C'est grâce à ce type de règles que les décideurs dans ces différents secteurs peuvent rendre une prévision plus précise et donc plus efficace. Mettons dans ce qui suit plus de lumière sur quelques-unes de ces applications intéressantes dont les résultats ont pu être améliorés grâce l'analyse des motifs séquentiels extraits, avec la considération particulière de celles que nous avons trouvé moins abordées dans la littérature relative.

3.2.1.Planification commerciale : Les motifs séquentiels véhiculent une excellente solution d'analyse de tendances et de comportements des consommateurs à travers le temps. Si par exemple, les séquences fréquentes d'achats de clients extraites à partir de la base de données d'un concessionnaire de véhicules, indiquent que les propriétaires d'un modèle précis de voiture changent leurs véhicules aux 36 mois, le concessionnaire pourra fidéliser sa clientèle en lui faisant parvenir un dépliant sur les nouveaux modèles quelques semaines avant ce délai. Ils apportent également une aide très intéressante dans le placement des articles achetés fréquemment dans un ordre particulier par le même client. Ce placement peut être dans une étagère, dans un catalogue, sur une page Web ou dans une publicité. Ainsi, les motifs séquentiels offrent la possibilité d'identifier des opportunités de ventes croisées et permettent d'améliorer le ciblage de consommateurs potentiels. Ils permettent également de définir des offres personnalisées en se basant sur les achats précédents des clients. L'essor du commerce par publipostage ou mailing qu'il soit électronique ou par poste et la diffusion d'outils de routage qu'ils soient logiciels, matériels ou techniques ont grandement facilité le développement d'entreprises spécialisées dans la vente par correspondance. Ces dernières qui effectuent des envois en nombre, se servent des motifs séquentiels afin d'améliorer la présentation de leurs articles, choisir un article à mettre en promotion, maximiser l'impact des publicités Online, réduire les coûts de poste en identifiant les acheteurs potentiels, et dans l'augmentation des taux de réussite de leurs mailings.

3.2.2.Recherche médicale : Les informations relatives au suivi des patients sont systématiquement stockées dans des bases de données dans la plupart des organismes médicaux (hôpitaux, cliniques, laboratoires d'analyses médicales, cabinets dentaires, …etc.). Les motifs séquentiels extraits à partir de ces bases de données sont d'extrême

utilité dans ce domaine. Ils fournissent à ces établissements des solutions leur permettant de mieux adapter les comportements sanitaires à la base de pathologies antérieures rencontrées, citons par exemple :[13]

L'aide au diagnostic médical par l'identification de symptômes précédant les maladies et dans certains cas, la définition de traitements en prédicant des symptômes ultérieurs.

L'aide à l'évaluation de l'efficacité de traitements ou de thérapies à long terme et la prédiction de l'impact physique et psychologique de thérapies lourdes sur les patients.

L'aide à la prédiction de résultats d'analyses médicales par la combinaison des caractéristiques des patients aux résultats d'autres analyses.

L'aide à l'analyse des risques sanitaires en identifiant les populations à risque vis-à-vis de certaines maladies par la prédiction de caractéristiques de populations pouvant être touchées par telle ou telle pathologie.

L'aide à l'analyse de la satisfaction des patients afin de mettre en évidence les sources de satisfaction ainsi que les leviers à actionner pour agir sur les causes d'insatisfaction.

L'aide à la prédiction des dépenses de santé par l'analyse de l'historique dépenses des années antérieures.

3.2.3.Recherche génomique L'acide désoxyribonucléique ou ADN est une molécule, retrouvée dans toutes les cellules vivantes, qui renferme l'ensemble des informations nécessaires au développement et au fonctionnement d'un organisme. C'est aussi le support de l'hérédité car il est transmis lors de la reproduction. Il porte donc l'information génétique et constitue le génome des êtres vivants. Cette information est vue par les spécialistes en génomique, comme un texte écrit dans un alphabet de 4 lettres (A, C, T, et G). Le nombre de lettres présentes dans le génome humain est de 3 milliards environs. Ce code de plusieurs millions de pages est maintenant entièrement transcrit et numérisé, et même disponible en accès libre sur internet. Celui d'autres espèces est en cours de décodage. De par cette forme enchaînée, ce code peut être facilement vu comme une séquence de caractères. La technique des motifs séquentiels est considérée comme la plus adaptée à l'analyse et la compréhension de message ,Son application à la recherche par exemple, de sous séquences fréquentes particulières, répétées, cycliques, rares ou inhabituelles et dans la reconstitution de séquences manquantes, ouvre de nouvelles perspectives à la recherche génomique, génétique, médicale et biologique .L'analyse du génome des plantes et insectes nuisibles ouvre de nouvelles perspectives dans le domaine de l'agriculture et de celui des espèces en voie de disparition, peut contribuer à la préservation de ces espèces.

3.2.4. Réseaux de télécommunication : Les compagnies de télécommunication génèrent et entretiennent d'énormes quantités d'informations au niveau de leurs bases de données. Ces informations incluent les données de détails d'appel, qui décrivent les appels qui traversent les réseaux de télécommunication, les données de réseau, qui décrivent l'état des composants matériels et logiciels dans le réseau, et les données clients, qui décrivent les profils de leurs abonnés. L'analyse de ces gigantesques ensembles de données par la technique des motifs séquentiels permet de découvrir des enchaînements de facteurs particuliers agissant sur la qualité de service et sur les performances des systèmes et réseaux de télécommunication. Citons quelques applications très intéressantes dans ce domaine :

Prédire des types d'incidents en se basant sur des enchaînements particuliers de défauts et identifier les causes d'anomalies par l'analyse de ces enchaînements.

Détecter les causes de situations alarmantes (par exemple, l'encombrement d'un ou de plusieurs points du réseau).

Analyser les erreurs de transmission dans les réseaux de télécommunication.

Des milliers d'alarmes sont détectées sur ces réseaux chaque jour : les motifs séquentiels peuvent être utilisés pour filtrer les alarmes non informatives.

Anticiper les défaillances coûteuses ou génératrices d'autres défaillances dans les équipements des installations de télécommunication.

D'autres applications sur les données clients et les données de détails d'appel montrent que les motifs séquentiels sont efficacement utilisés dans la détection de fraudes de télécommunications par l'identification de séquences particulières d'appels.

3.2.5. Maintenance industrielle [14] : Les motifs séquentiels par leur précision peuvent fournir aux directions de recherche et de développement des entreprises industrielles, des solutions leur permettant de perfectionner leurs procédés de fabrication et mieux adapter leurs plans de maintenance. Leur utilisation dans ces procédés, notamment dans le diagnostic des équipements de production permet de mieux prévenir les pannes et minimiser les défauts. La maintenance anticipée est une application typique dans ce domaine. En effet, l'analyse des enchaînements des opérations de maintenance effectuées sur ces équipements, permet souvent d'anticiper des types de pannes particulières, coûteuses ou engendrant d'autres défaillances.

3.2.6. Banques et Assurances : Les données de banques et assurances sont riche d'informations utiles sur leurs clients. De plus, leurs bases de données sont initialement

conçues pour enregistrer automatiquement pour chaque nouvelle transaction insérée son temps valide, indiquant par exemple, sa date et son heure exacte (voir, identifiant temporel, Section 3). Les motifs séquentiels s'adaptent particulièrement à l'analyse de telles informations et obtiennent de très bons résultats :

Améliorer le ciblage des clients : Grâce aux motifs séquentiels comportementaux, décrivant des enchaînements de transactions effectuées par ses clients ayant obtenus un crédit personnel, une banque peut cibler parmi tous ses clients, ceux qui ne l'ont pas déjà et qui pourraient être intéressés par un tel crédit, ce qui lui permet à de minimiser les coûts d'une telle opération et augmenter le taux de réussite de son mailing.

Minimiser le risque client : Les motifs séquentiels identifiant des séquences de conduites particulières ou inhabituelles, peuvent permettre par exemple, la description d'une conjonction de paramètres de la forme (catégorie client, opération demandée, montant, délais, …etc.) servant avec ces motifs à l'identification de clients à risque lors de l'octroi d'un crédit au niveau d'une banque ou à la souscription d'un contrat d'assurance.

Anticiper les incidents financiers : Les séquences de types de transactions se finissant fréquemment par un incident financier offrent la possibilité d'anticiper la survenue de tels incidents dans le cycle de vie d'un client. Ceci permet à la banque de limiter le risque de perte finale de la créance ainsi que le risque de perte de clients solvables.

Détection de fraudes : Les séquences de caractéristiques de retraits par carte bancaire fréquentes dans l'ensemble de ces séquences se terminant par un incident de carte, peuvent être considérées a priori comme des utilisations frauduleuses de cartes bancaires. Ce même principe appliqué aux données d'assurances, permet de découvrir les fausses déclarations de dégâts ainsi que les assurés suspects !

3.2.7. Applications Internet : De très grandes quantités de données internet, sans cesse croissantes, comme les adresses IP des utilisateurs et les URL demandées, sont automatiquement récupérés par les serveurs Web et stockées dans des fichiers Log. Ces informations riches sur la dynamique du Web, sont de grande valeur pour le Web Mining qui consiste à analyser les sites Web en fonction de leurs usages et parcours par les utilisateurs (Web Usage Mining) mais également en fonction de leurs contenus et structures (Web Content Mining).

La structure hiérarchique et l'enchaînement de liens de pages d'un site Web peuvent être facilement représentés par un graphe. Les chemins de navigation relatifs aux parcours Web des utilisateurs représentent en fait des enchaînements de liens de pages

de sites Web. Les motifs séquentiels s'adaptent très bien à ce type de représentation facilement transformable en séquences de liens URL et adresses IP. C'est pourquoi, les applications internet de ces motifs deviennent de plus en plus nombreuses et pratiques. Résumons dans ce qui suit quelques-unes de ces applications intéressantes :[15]

L'analyse des parcours des utilisateurs d'un site Web par la détection de pages fréquemment visitées dans un ordre particulier permet d'améliorer la conception du site par la réorganisation de ses pages et liens. Les pages concernées seront par exemple, reliées de manière hypertextuelle si ce n'est pas déjà le cas.

En se basant sur les séquences fréquentes de pages visitées et les profils de ses utilisateurs, un site Web peut ajuster dynamiquement la présentation de ses pages et liens, afin de positionner automatiquement l'information que le visiteur cherche ou encore pour faire des recommandations à un utilisateur (on parle déjà de sites adaptatifs !). Les motifs séquentiels côté client permettent d'optimiser le cache du navigateur et d'effectuer du perfetching (préchargement).

E−commerce : Les motifs séquentiels sur les tendances des consommateurs internet, obtenus par l'analyse de l'historique des chemins de navigation à partir d'un ou plusieurs serveurs Web, permettent d'extraire des relations entre objets présentés servant à maximiser l'impact des publicités Online. Ils permettent également d'améliorer le ciblage de clients potentiels et mesurer l'impact des opérations de Marketing−Web, …etc.

Sécurité de l'information : La technique des motifs séquentiels est très efficace dans la détection de failles dans la sécurité d'accès à l'information Web et dans l'identification des accès non autorisés (par l'analyse de séquences fréquentes de ces accès).

3.2.8. Analyse de données spatiales

De nombreux domaines utilisent des représentations spatiales leur permettant de stocker et interroger des descriptions géométriques 3D. C'est le cas par exemple, de la géographie, la géologie, le génie civil, l'architecture et la robotique. D'autres domaines comme la gestion du trafic routier, l'accidentologie, les utilisations militaires et la météorologie sont des exemples d'applications nécessitant le support de données spatio−temporelles. De plus, les objets représentés ont, en plus de la composante spatiale (forme, dimensions, …etc.) ou spatio−temporelle (forme, dimensions, position, direction, vitesse, …etc.), une description par des attributs classiques comme (couleur, température, poids, …etc.)

Les motifs séquentiels extraits depuis les données relatives aux objets définissent généralement des relations d'enchaînement entre objets sur la base de leurs caractéristiques spatiales et non spatiales. Ces motifs sont utilisées pour l'aide à la prédiction d'évènements naturels (éruptions, tremblements de terre, ouragans, …etc.) ainsi que dans l'aide à la prévision météo. La prédiction de goulots d'étranglement dans la circulation routière est une autre application très intéressante de ces motifs.

3.3. Concepts Généraux

Comme mentionné en introduction de cette section, les bases de données relatives à la problématique des motifs séquentiels comportent pour chaque transaction, en plus de son ensemble d'items, un identifiant temporel indiquant le temps valide de la transaction, comme par exemple sa date ou son heure. C'est pourquoi, ces bases de données sont souvent qualifiées de bases de données temporelles ou historisées. Toutefois, la problématique des motifs séquentiels peut être perçue comme l'extension de celle des règles d'association. Elle est présentée dans [16] de la manière suivante (nous gardons les concepts de clients et d'achats tout en les adaptant par la suite aux types de données relatives à notre application)

Définitions

Définition 08(Transaction) : Une transaction pour un client C est un triplet (CID, Date, Itemset) formé de l'identificateur unique du client, de la valeur de l'identifiant temporel pour cette transaction et de l'ensemble des items de la transaction, représentant tous les items achetés par C à cette même date.

Définition 09(Base de données transactionnelles)

Une base de données temporelle D est un ensemble de transactions (CID, Date, Itemset). }}

D ={T :(TID,Date,Itemset) / Itemset ={a∈ I}}

Exemple

CID	DATE	ITEMSET
C_1	01/01/2008 02/01/2008 04/01/2008 18/01/2008	{B,F} {B} {C} {H,I}
C_2	11/01/2008 12/01/2008 29/01/2008	{A} {C} {D,F,G}
C_3	05/01/2008 12/02/2008 18/02/2008	{C,E,G} {A,B} {I}
C_4	06/02/2008 07/02/2008	{B,C} {D,G}

Table 2.2 – Base de données temporelle ordonnée Par CID et par date de transaction.

Ce tableau représente une base de données temporelle ordonnée selon L'identifiant unique du client CID et l'identifiant temporel Date de cette base. La transaction effectuée par le client C1à la date du 18/01/2008 est vue comme un triplet (C1 18/01/2008, {H, I}).Les deux items H et I étant été achetés lors de cette même transaction.

Définition10(Séquence) : Une séquence de données est une liste ordonnée non vide d'itemsets Si, notée S =<s1s2…sn> avec $i \in [1…n]$ indique l'ordre d'apparition de si dans S .

Une séquence de données peut être alors vue, comme une liste de transactions respectant une relation d'ordre temporel. Mentionnons que l'indice i indique uniquement l'ordre d'apparition des itemsets si et n'est pas une mesure de temps absolu.

Exemple

Reprenons la base de données du Tableau 04 La liste ordonnée des trois transactions effectuées par le client C2 est donnée par la séquence avec : s1={A}, s2={C} et s3={D,F,G}. Cette séquence se lit de la manière suivante : le client C2 a acheté l'article A, puis l'article C, puis simultanément les trois articles D, F et G.

Il est important de mentionner que dans la problématique des motifs séquentiels, la notion de simultanéité ou de succession peut être interprétée différemment selon les contextes applicatifs : les achats peuvent être considérés comme simultanés s'ils ont été effectués lors de la même transaction ou pendant une période de temps bien spécifiée (semaine, mois, …etc.). Ce type d'interprétation est rendu possible grâce aux paramètres temporels qui peuvent être spécifiés sur ces motifs comme nous le verrons prochainement.

Définition 11(Sous séquence contiguë)

On dit que la séquence S'= <s'1s'2…s'n>est une sous séquence contiguë de la séquence S== <s1s2…sn, si l'une des conditions suivantes est vérifiée :

(1)S' peut être dérivée de S en éliminant un ou plusieurs items de s1, s2,…, sm.

(2) S' peut être dérivée de S en éliminant un item de si et si est constitué d'au moins deux items avec $i \in [2, (n-1)]$.

(3) S' est sous séquence contiguë d'une séquence S'' avec S'' sous séquence contiguë de S.

Exemple

Les séquences , <{B , {C,D}{E}> ,<{A,B}{C}{E}{F}>,<{C}{E}> et<{B}{C}{E}> et sont toutes des sous séquences contiguës de la 6−séquence <{A,B}{C,D}{E}{F}>. . Toutefois, les séquences <{A,B}{C,D}{F}>,<{A}{E}{F}>et <{B}{C}{F}> , et ne le sont pas.

Définition 12(Séquence client)

La séquence client du client C dans une base de données D est définie comme la liste des itemsets de toutes les transactions de C dans D, ordonnés selon l'identifiant temporel de D.

$S(C) = <s1s2 ...sn> \forall i \in [1,n], \exists (C,Datei,sj) \in D, sj \subseteq si \wedge \forall I,j \in [1,n], I < j \Rightarrow Datei < Datej$

Avec: $n \leq \{T : (CID,DATE,Itemset) \in D/T(CID)=C\}$

Nous comprenons facilement que la liste ordonnée de toutes les transactions d'une base de données relatives aux achats d'un client donné, peut être vue comme une seule séquence pour ce client. De là, toute base de données temporelle D peut être transformée en regroupant pour chaque client toutes les transactions effectuées par ce

client dans une seule séquence. Le critère d'ordre dans cette transformation est sans doute l'identifiant temporel de D. Le résultat est une base de séquences clients notée D', comme le formalise la définition suivante.

Définition 13 (Inclusion) : une séquence S'= <s'1 s'2…..s'n> est une sous-séquence de S=<s1 s2 ….sn> s'il existe des entiers a1 < a2<….<aj<….<am tels que s'1 ⊆sa1, s'2 ⊆ sa2,....., s'm ⊆ sam. On dit que S'est incluse dans S.

Exemple : La séquence. S'=<(a)(b, c)(d)> est incluse dans la séquence S=< (a, d, e)(g, h)(f)(b, c, e)(d, e, f) > car (a) ⊆ (a, d, e), (b, c) ⊆ (b, c, e) et (d) ⊆ (d, e, f). En revanche, <(a)(b)> 6 ⊆/< (a, b)> (et vice versa). Les deux séquences < (a)(b)> et < (a, b) > sont dites incomparables.

Définition 14(Longueur d'une séquence)

La longueur d'une séquence S est le nombre d'un d'items dans cette séquence. Une séquence de longueur K est une K-séquence..

Exemple

La séquence< {A}{C}{D}{C, E} > est une 5 −séquence, même si cette dernière contient seulement4 itemsets. L'item C est contenu dans deux transactions et est donc compté 2 fois.

Definition15 (Base de séquences de données)

Une base de séquences clients, notée D', transformée d'une base de données temporelle Dest définie comme l'ensemble de toutes les séquences clients présentes dans D.

$$D' = SEQ(D) = \{S(C_i), i = 1..\|D\| / \|D\| = nombre\,de\,clients\,dans\,D\}$$

Avec:

S(Ci) =<s1s2...sn>∀i∈[1,n], ∃(Ci,Datej,sk) ∈D,sk⊆sj∧∀j ,k∈[1,n],j <k⇒Datej<Datek

Avec: n≤{T :(CID,DATE,Itemset) ∈D/T(CID)=Ci}

Exemple

CID	*SEQUENCES DE DONNEES*
C_1	<{B,F} {B} {C} {H,I}>
C_2	<{A} {C} {D,F,G}>
C_3	<{C,E,G} {A,B} {I}>
C_4	<{B,C} {D,G}>

Table 2.3 – Base de séquences de données obtenue par transformation de la base de données du Tableau 02.

Le Tableau représente une base de séquences de données obtenue par la transformation en base de séquences de la base de données de la table 02. Il est important de mentionner qu'une grande part des algorithmes de recherche de motifs séquentiels effectue cette transformation avant de procéder à l'extraction proprement dite de ces motifs.

Définition 16 (Fréquence d'une séquence): Une séquence est considérée fréquente, si le support de cette séquence est supérieur ou égal au support minimum. Celui-ci est introduit par le client.

Définition 17(Support d'une séquence)

On dit qu'une séquence client S, dans une base de séquences de données D', supporte une séquence quelconque S', si S'est sous séquence de S. Le support de S' dans D', noté Sup(S'), est donné par le nombre de séquences clients de D' qui supportent S'.

$Sup(S) = \{s: \langle s1\ s2\ ...sn\rangle \in D' / S \subseteq s\}$

Exemple

Considérons la base de séquences de données du Tableau 4 La séquence de données <{C}{I}>est supportée par les deux clients :C1 et C3, son support est donc égal à 2. Le même support pour la séquence <{C}{D,G}>qui est supportée par C2 et C4.

Il est clair qu'une séquence client n'est considérée qu'une unique fois dans le calcul du support d'une séquence donnée, autrement dit, qu'elle peut contenir plusieurs fois cette séquence sans que le calcul de son support ne tienne compte du nombre de ses apparitions dans la séquence client. Il lui suffira donc de s'assurer seulement, que cette séquence est contenue dans la séquence client, pour en augmenter le support.

Définition 18(Séquence fréquente maximale)

Une séquence S est dite fréquente dans une base de séquences de données D' si et seulement si son support dans D'est au moins égal au seuil de support minimum défini par l'utilisateur. La séquence S est dite fréquente maximale dans D' si et seulement si elle est fréquente dans D' et n'est sous séquence d'aucune autre séquence fréquente dans D'.

Fréquente Maximale$\Leftrightarrow$ Sup(S) $\geq$MinSup$\wedge/\exists$ S'$\supset$S/ Sup(S') $\geq$MinSup

Exemple

Considérons la base de séquences de données du Tableau 3 et un seuil de support minimum égal à 2/4, autrement dit, 50%. Pour qu'une séquence soit retenue fréquente, il suffira donc qu'au moins deux clients dans cette base supportent cette séquence. Les séquences fréquentes maximales sont alors les suivantes : <{B} {I}>, <{C}{ I}> et <{C} {D,G}. Les deux premières apparaissent dans les séquences de données des clients C1 et C3, alors que la dernière apparaît dans les séquences de données des clients C2 et C4.

Cette notion de séquence fréquente maximale est fondamentale dans la problématique des motifs séquentiels. En effet, elle définit le problème à résoudre par toute approche visant la découverte de ces motifs dans une base de données : extraire à partir de sa transformée en base de séquences de données, l'ensemble FM de toutes les séquences fréquentes maximales, appelées règles d'association séquentielles ou motifs séquentiels, autrement dit, l'ensemble :

FM= {S:$\langle$ s1 s2 ...sm$\rangle$/ Sup(S) $\geq$MinSup$\wedge/\exists$ S'$\supset$S/ Sup(S') $\geq$MinSup

avec s1 ,s2 ,...,sm$\subseteq$ I $\wedge$ s1 ,s2 ,...,sm$\neq\phi$

3.4.Paramètres associés aux motifs séquentiels

La recherche de motifs séquentiels dans une très grande base de données ou lorsque le seuil de support minimum spécifié par l'utilisateur est trop faible, peut conduire à un nombre très important de ces motifs, conduisant à des phases de post-traitement coûteuses ou imposant à l'utilisateur de fouiller (encore !) dans ces motifs. Afin de mieux cibler les connaissances extraites, la recherche peut être restreinte à seulement ce qui intéresse l'utilisateur. Ceci limite le nombre de motifs inutilement extraits et peut potentiellement réduire les temps d'extraction, à condition que cette restriction soit traitable au cours de la phase de recherche.

Plusieurs techniques ont été proposées afin de prendre en compte un certain nombre de contraintes de restriction, en plus de la contrainte de support minimum. Plusieurs types de ces contraintes ont pu être définis. Toutefois, nous nous intéressons au type de contraintes dites temporelles, qui comme que nous le verrons dans ce mémoire, nous seront de très grande utilité pour notre application à la maintenance industrielle. Elles sont généralement spécifiées par l'utilisateur sous forme de paramètres, et c'est pour cette raison qu'elles sont souvent dites **paramètres temporels.**

La définition de paramètres temporels nécessite l'introduction de la notion de transaction généralisée et par conséquent celle de motifs séquentiels généralisés. En effet, l'interprétation de transaction utilisée jusqu'à présent présente une rigidité pour certaines applications : si l'intervalle de temps entre deux transactions est suffisamment court, on pourrait envisager de les considérer comme simultanées. A l'inverse, deux évènements trop éloignés peuvent ne pas avoir de lien entre eux ou ne pas intéresser l'utilisateur.

Cette problématique a été introduite par [17] qui propose une généralisation de la notion de transaction par la considération de paramètres temporels sur les motifs séquentiels. L'utilisateur pourra alors choisir d'imposer :

→ **Une durée temporelle minimale :** à respecter entre deux itemsets consécutifs d'une séquence, ce qui permettra de considérer des itemsets comme trop proches pour appartenir.

→**Une durée temporelle maximale :** à respecter entre deux itemsets consécutifs d'une→ séquence, ce qui permettra de considérer des itemsets comme trop éloignés pour appartenir à une même séquence.

→**Une fenêtre temporelle** :par laquelle deux itemsets de deux transactions différentes mais de dates proches peuvent être considérés comme simultanés, ce qui permettra de les regrouper dans un même itemset.

3.4.1. L'intervalle de temps [MinGap, MaxGap]

L'intervalle de temps [MinGap, MaxGap] définit la distance temporelle minimale et maximale à vérifier entre chaque deux itemsets consécutifs dans une séquence de données lors du calcul de son support par le processus de recherche de motifs séquentiels. Grâce à cette notion, l'utilisateur peut spécifier un intervalle de temps plus ou moins défini comme c'est le cas pour l'intervalle [0,10] jours, pour ne considérer que les séquences l'intéressant en respectant cette distance temporelle entre leurs itemsets consécutifs.

3.4.2. La fenêtre temporelle [WinSize]

La fenêtre temporelle (Slidingwindow) également appelée fenêtre d'évènements (Event foldingwindow) définit la distance temporelle minimale pour que deux événements soient considérés comme faisant partie de deux transactions distinctes. La fenêtre temporelle donne alors une plus grande flexibilité à la définition de transaction, en offrant la possibilité de considérer comme simultanées des transactions différentes, qui se déroulent à l'intérieur de la même fenêtre temporelle. Par exemple, lorsque cette fenêtre est de sept jours (WinSize = 7), un item acheté le 03/03/2008 et un autre le 08/03/2008 sont considérés comme simultanés, car ils se sont produits à l'intérieur de cet espace temporel.

Exemple

Considérons le Tableau 2.4 qui représente une base de données temporelle réduite à un seul client. La séquence de données du client C1 est donc S1 = = <{A}{B,C}{D}{E,F}{G}>..

CID	*DATE*	*ITEMSET*
C_1	01/02/2008	{A}
	02/02/2008	{B,C}
	03/02/2008	{D}
	04/02/2008	{E,F}
	05/02/2008	{G}

Table 2.4 – Base temporelle réduite a un seule client

Considérons une séquence candidate S, nous utiliserons trois possibilités différentes d'affectations de valeurs pour S, MinGap, MaxGap et WinSize afin d'illustrer l'effet des paramètres temporels sur la recherche de motifs séquentiels.

(1) S =<{A,B,C,D}{E,F,G}>, WinSize = 3, MinGap = 0 et MaxGap = 5

Comme supportée par S1. En effet, le paramètre WinSize permet de regrouper les itemsets {A}, {B,C} et {D} ainsi que les itemsets {E,F} et {G} afin d'obtenir {A,B,C,D} et {E,F,G}. De plus, la contrainte MinGap entre les itemsets {D} et {E,F} est respectée.

S = <{A,B,C,D D}{G}>, WinSize = 1, MinGap = 3 et MaxGap = 4

Il n'y a aucun moyen de regrouper des itemsets de S1 de manière à accepter S comme séquence supportée par S1. En effet, le paramètre WinSize = 1 ne permet pas de regrouper les itemsets {A}, {B,C} et {D}.

S = < {A,B, C}{F,G}>, WinSize = 2, MinGap = 3 et MaxGap = 4

Il n'y a qu'un seul moyen d'obtenir les deux itemsets {A,B ,C} et {F,G} via WinSize en regroupant t {A} et {B,C} puis {E,F} et {G} mais dans ce cas, MinGap n'est plus respecté entre ces deux itemsets car ils sont espacés de seulement 2 jours alors que la contrainte MinGap exige 3 jours minimum.

La fenêtre temporelle WinSize comme unité de temps dans les motifs séquentiels

Il est important de mentionner que la borne supérieure MaxGap de l'intervalle de temps, ne peut être spécifiée inférieure à la fenêtre temporelle WinSize, car celle-ci défini l'unité fondamentale de temps d'une transaction généralisée. Soulignons que si l'intervalle de temps est spécifié [0,0] par l'utilisateur, seules les associations entre items survenus dans cette fenêtre seront considérées, comme c'est le cas dans les motifs fréquents classiques.

Quant à la recherche de motifs séquentiels sous contraintes, il est clair que la Définition (support d'une séquence) qui donne le support d'une séquence dans une base de données devient insuffisante dans ce cas. Elle doit être généralisée à l'intégration des différents paramètres temporels spécifiés par l'utilisateur, afin que cette tâche puisse les vérifier dans le calcul des supports des séquences considérées, ce qui est donné par la Définition suivante.

Definition2.17' (Support d'une séquence sous contraintes temporelles)

On dit qu'une séquence client S, dans une base de séquences de données D', supporte une séquence S' s'il existe une sous séquence S'' de S contenant S' et respectant dans D' l'ensemble des paramètres temporels spécifiés par l'utilisateur. Le support de S' dans D' est donné par le nombre de séquences clients de D' qui supportent S'.

En somme, la spécification de contraintes temporelles sur les motifs séquentiels permet non seulement de filtrer ces derniers pour ne rapporter à l'utilisateur que les motifs l'intéressant mais également de les enrichir, en les rendant plus souples, plus précis et plus adaptés à types d'applications de plus en plus exigeantes.

4.Processus d'extraction de motifs séquentiels

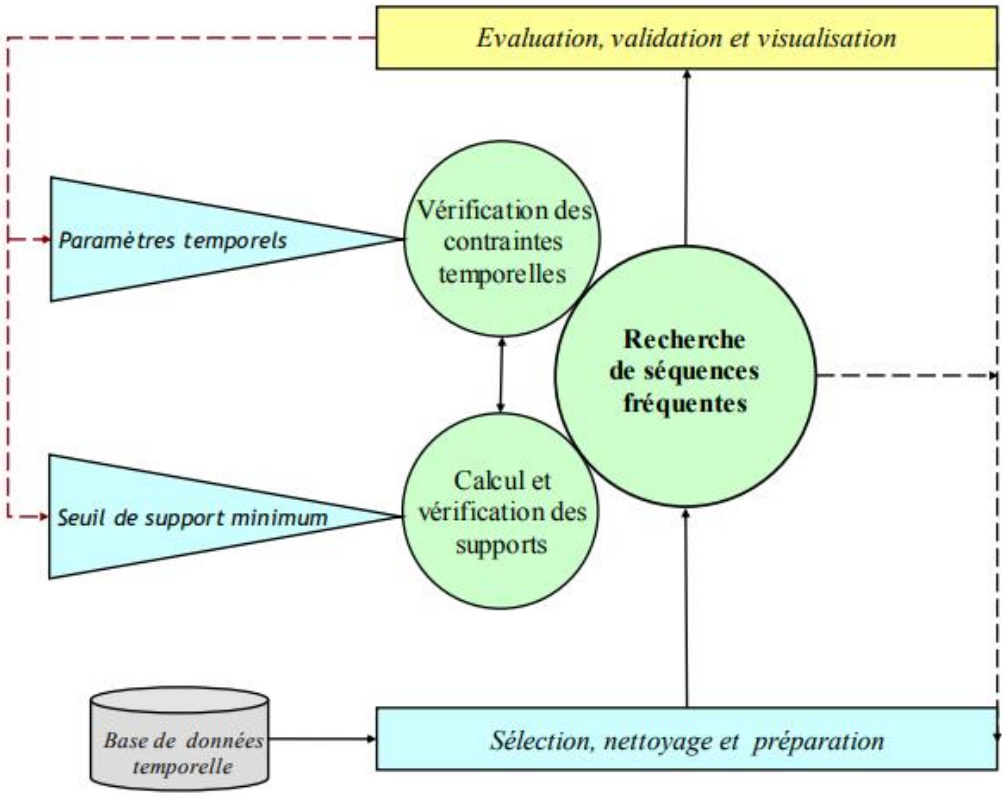


Figure 2.1– Processus ECD adapté à l'extraction de motifs séquentiels généralisés

L'extraction de motifs séquentiels dans une base de données temporelle consiste à découvrir l'ensemble de toutes les séquences fréquentes maximales dans sa transformée en base de séquences de données. Le processus aboutissant à la réalisation de cet objectif ne se limite pas à une extraction automatique, c'est un processus itératif et interactif constitué de plusieurs phases, allant de la sélection et la préparation des données d'analyse jusqu'à la visualisation des résultats, en passant par la phase d'extraction proprement dite de ces motifs, phase centrale de ce processus, comme le montre la Figure .

4.1. Sélection et préparation des données

Cette phase consiste à sélectionner les données (attributs et objets) utiles à la tâche d'extraction de motifs séquentiels. Les données sources résident souvent dans une base de données temporelle. Il est très important de mentionner que l'attribut de la base de données jouant le rôle de l'identifiant temporel de l'analyse, ne peut en aucun cas être ignoré, car sans cet identifiant aucune extraction de motifs séquentiels ne pourra être effectuée. L'extraction de motifs séquentiels peut être généralement appliquée à divers

types de bases de données : relationnelles, orientées objets, spatiales, multi−média, textuelles], …etc.

La sélection et la préparation des données est une étape indispensable pour tout processus de recherche de motifs séquentiels, afin qu'il soit possible d'appliquer les algorithmes d'extraction sur des données de natures différentes provenant de sources différentes et de concentrer la recherche sur les données utiles. En partant du principe que de la qualité des données en entrée que dépend la qualité des résultats, cette phase permet de n'extraire que les informations susceptibles d'intéresser l'utilisateur. De plus, la taille des données est réduite ce qui permet de minimiser les temps d'extraction et assurer une meilleure efficacité.

4.2. Recherche de séquences fréquentes

Comme le montre la Figure2.1, cette étape représente la phase centrale du processus d'extraction de motifs séquentiels dans une base de données. En effet, elle consiste à extraire à partir de sa transformée en base de séquences de données consolidée et préparée lors de la phase précédente, l'ensemble de toutes les séquences fréquentes maximales d'items ou de groupes d'items dans cette base, autrement dit, celles vérifiant le seuil de support minimum défini par l'utilisateur tout en étant non contenues dans aucune autre séquence fréquente.

La Définition (Séquence fréquente maximale)est la définition claire, utile et fondamentale dans la description de cet objectif de toute tâche d'extraction de motifs séquentiels. Le calcul des supports des différentes séquences de données considérées pendant la phase de recherche de séquences fréquentes, appelées séquences potentiellement fréquentes ou séquences candidates, est réalisé en se basant sur la Définition (support d'une séquence), qui donne le support d'une séquence. Dans la mesure où l'utilisateur peut spécifier d'autres types de contraintes autres que le seuil de support minimum, comme les contraintes temporelles (intervalle de temps et fenêtre temporelle), la Définition ((Support d'une séquence sous contraintes temporelles)qui constitue la généralisation de cette dernière à cette considération doit être celle utilisée.

Enfin, il est important de mentionner que l'extraction de motifs séquentiels dans une base de données est un problème non trivial et peut être très coûteuse en terme de temps de réponse et de consommation mémoire car le nombre de séquences candidates à considérer pendant cette phase, dépend exponentiellement du nombre d'items et de la densité de la base de données, du seuil de support minimum et d'éventuelles contraintes temporelles spécifiées par l'utilisateur, comme nous le verrons dans le prochain chapitre.

4.3.Evaluation et visualisation des motifs séquentiels

Cette phase représente l'ultime étape du processus d'extraction de motifs séquentiels. Elle est généralement constituée de ces trois tâches importantes : l'évaluation, qui mesure la fiabilité des motifs séquentiels extraits, puis la validation des motifs intéressants par les experts du domaine et enfin, la présentation des résultats à l'utilisateur sous une forme actionnable.

La présentation des résultats consiste à visualiser par l'utilisateur l'ensemble de toutes les séquences fréquentes maximales valides et leur interprétation afin d'en déduire des connaissances utiles pour l'activité concernée. Ainsi l'expert du domaine peut juger de leur pertinence et utilité. Mais le nombre souvent considérable de ces motifs impose le développement d'outils de classification de motifs selon leurs propriétés, de sélection de sous ensembles de motifs selon des critères définis par l'utilisateur et de visualisation de ces motifs sous une forme claire et intelligible. La forme de présentation peut être textuelle, graphique ou bien une combinaison de ces deux formes intelligibles.

Le développement d'un mode de visualisation et d'un mode opératoire adaptés à l'activité concernée et aux attentes des utilisateurs, peut être facilité à l'aide d'outils de visualisation de connaissances (Knowledge Visualization Tools). Ces derniers font généralement intervenir des notions d'interaction homme−machine, de convivialité et de visualisation. Du fait de l'importance accordée de plus en plus au rôle de l'utilisateur, ils sont devenus des composantes majeures dans toute bonne application d'extraction des connaissances.

Le dernier point important concerne les connaissances de l'expert humain, qui sont essentielles lors des phases de prétraitement afin d'assister la sélection et la préparation des données et de post-traitement, pour l'interprétation et l'évaluation des motifs séquentiels extraits. En fonction de l'évaluation de ces motifs, les paramètres utilisés lors des phases précédentes (critères de sélection et de préparation des données, seuil de support et paramètres temporels) peuvent être modifiés avant d'effectuer à nouveau l'extraction, comme le montre la Figure2.1 , ce qui permet souvent, d'améliorer la qualité des résultats.[04]

5.Algorithme général d'extraction de motifs séquentiels [07]

Après le développement de nombreux algorithmes d'extraction d'itemsets fréquents pour la génération de règles d'association, tels que par exemple, l'algorithme pionnier APRIORI et ses diverses alternatives, les chercheurs dans le domaine du Data mining

se sont vite orientés vers l'application du même principe à l'extraction de motifs séquentiels. Toutefois, il s'est avéré que le nouveau problème est beaucoup plus complexe, car l'ordre des items, comme nous l'avons évoqué tout au long de ce chapitre, est dans ce cas une propriété fondamentale à l'inverse des ensembles fréquents et par conséquent, plusieurs optimisations identifiées pour traiter ces ensembles ne se transposent pas aux séquences d'items.

Nous présentons dans cette section la démarche générale permettant l'extraction de motifs séquentiels dans une base de données temporelle. Il s'agit en fait du premier algorithme dans ce domaine, APRIORI−ALL, introduit par Agrawal et Srikant dans [18] comme une adaptation de leur algorithme APRIORI traitant les ensembles fréquents, aux motifs séquentiels. Nous proposons alors d'expliquer ses différentes phases afin de mieux situer le problème et préparer l'analyse, pour le prochain chapitre, des différentes grandes approches d'extraction de motifs séquentiels, comme nous le verrons.

Principe de base

Étant donné une base de données temporelle D, considérons D' sa transformée en base de séquences de données, le problème consiste alors, à trouver l'ensemble de toutes les séquences fréquentes maximales dans D

Afin de réaliser cette tâche, APRIORI−ALL se base essentiellement sur la propriété d'antimonotonicité existante entre les séquences de données afin d'optimiser son espace de recherche. Cette propriété d'inclusion est en fait utilisée à chaque itération de l'algorithme afin de réduire, le nombre de séquences candidate considérées

Propriété 1 : (Monotonicité du support)

Soit D' une base de séquences de données et S1, S2 deux séquences de données avec S1 sous séquence contiguë de S2, alors, le support de S1 dans D' ne peut être strictement inférieur à celui de S2 dans D'.

$\forall$S deux séquences de données $S1 \subseteq S2 \Rightarrow Sup\ (S1) \geq Sup\ (S2)$

Cette propriété montre aisément que toutes les sous séquences contiguës d'une séquence fréquente sont aussi fréquentes. Elle est justifiée par le fait que toute séquence de données supportant une séquence, supporte forcément toutes ses sous séquences contiguës.

Propriété 2 :(Antimonotonicité du support)

Soit D' une base de séquences de données et S1, S2 deux séquences de données avec S1 sous séquence contiguë de S2, si le support de S1 dans D' est strictement inférieur au seuil de support minimum défini par l'utilisateur alors le support de S2 dans D' l'est également.

$\forall$S deux séquences de données S1$\subseteq$S2 $\wedge$ Sup(S1) <MinSup$\Rightarrow$ Sup(S2) <MinSup

Cette propriété montre aisément que toutes les sur-séquences contiguës d'une séquence non fréquente sont aussi non fréquentes. C'est une conséquence de la première. En effet, d'après celle-ci Sup(S2)$\leq$ Sup(S1) <MinSup, d'où Sup(S2) <MinSup et S2 ne peut être fréquente.

La propriété d'antimonotonicité est très intéressante pour tout algorithme d'extraction de motifs séquentiels puisqu'elle permet de ne considérer pendant la phase de recherche, aucune sur-séquence contiguë d'une séquence trouvée non fréquente, car selon cette propriété, elle ne pourra être fréquente. Autrement dit, elle permet de réduire l'espace de recherche de séquences fréquentes en ignorant à l'avance les séquences non fréquentes.

APRIORI−ALL se base sur ce principe et une recherche itérative par niveau pour générer les séquences fréquentes. Il s'agit de rechercher ces séquences par longueur croissante. Pour chaque longueur de séquence k, l'algorithme génère l'ensemble des séquences potentiellement fréquentes Ck en ne considérant que les séquences n'ayant pas de sous séquences contiguës non fréquentes, puis sélectionne dans Ck, celles qui sont suffisamment supportées dans la base de séquences de données D', c'est-à-dire, celles ayant le seuil de support minimum spécifié par l'utilisateur. Un parcours de D' est donc effectué pour calculer le support de chaque séquence candidate. Si des séquences candidates sont fréquentes, alors elles formeront l'ensemble des séquences fréquentes de longueur k noté Fk et serviront d'amorce pour générer l'ensemble Ck+1 des séquences candidates de longueur supérieure (k+1).

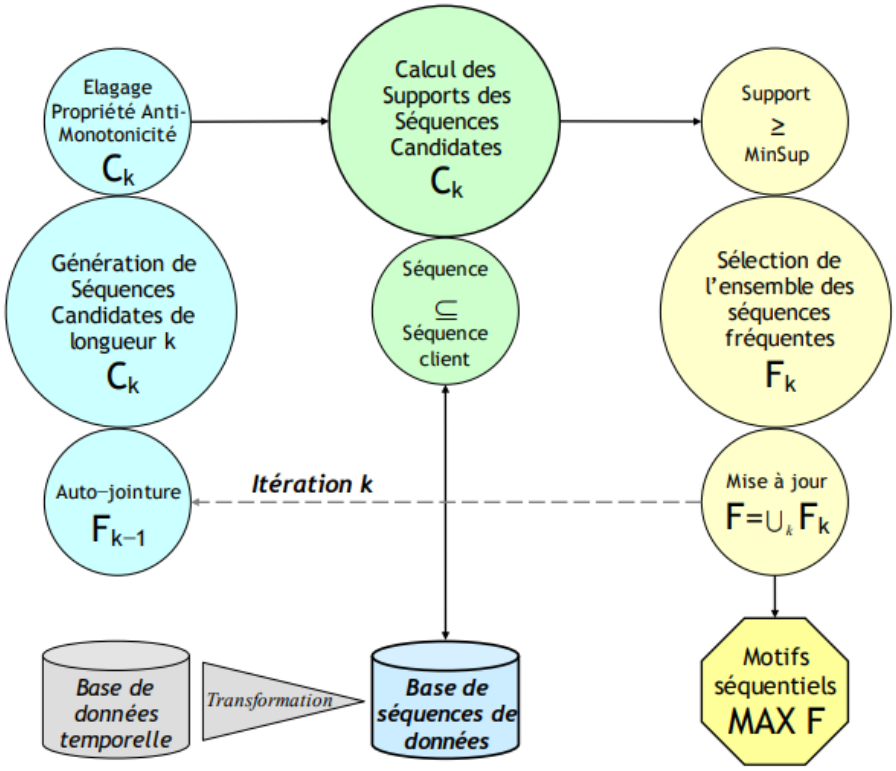


Figure2.2 – Principe de base pour l'extraction de motifs séquentiels dans une base de données temporelle.

L'algorithme réitère jusqu'à ce qu'aucune séquence candidate ne puisse être générée ou lorsque aucune séquence fréquente ne puisse être identifiée. L'union des Fk constitue alors l'ensemble F de toutes les séquences fréquentes trouvées. Cet ensemble est ensuite réduit afin de ne garder que les séquences fréquentes maximales, c'est-à-dire, celles n'étant contenues dans aucune autre séquence fréquente, et c'est ce dernier noté FM, qui représente l'ensemble des motifs séquentiels découverts par APRIORI−ALL, qui est retourné en résultat.

Algorithme 2.1 : *APRIORI−ALL* (*D'*, *MinSup*)
Extraction de motifs séquentiels dans une base de séquences de données *D'*, transformée d'une base de données temporelle, avec unseuil de support minimum *MinSup*.

Entrée : Base de séquences de données *D'*; Liste d'attributs I; Support minimum *MinSup*;
Sortie : Ensemble des séquences fréquentes maximales F_M ;

1. $F_1 = \{s:<a> / a \in I \wedge Sup(s) \geq MinSup\}$; // Séquences fréquentes de longueur 1
2. **Pour** (k=2; $F_{k-1} \neq \emptyset$; k++) **faire**
    3. **Début**
        4. $C_k = Gen-cand(F_{k-1})$;
        5. *Compter−supports*(C_k);
        6. $F_k = \{s \in C_k / Sup(s) \geq MinSup\}$;
    7. **Fin;**
8. $F = F_1 \cup F_2 \ldots \cup F_k$; // Ensemble des séquences fréquentes dans *D'*

9. **Retourner** $F_M = \{s \in F / \exists / s' \supset s, \text{Sup}(s') \geq MinSup\}$. // Séquences fréquentes maximales

Fonction : *Gen−cand*(*Fk*−1) − Génération de séquences candidates de longueur *k*

1. $C_k = \emptyset$;
2. **Pour chaque** séquence $s_1 \in F_{k-1}$ **faire**
3. **Pour chaque** séquence $s_2 \in F_{k-1}$ **faire**
4. **Si** $s_1[2..k-1] = s_2[1..k-2]$ **alors**
5. **Début**
6. $s = s_1 \otimes s_2$; // jointure entre s_1 et s_2, $s = s_1[2..k-1]+s_2[k-1]$
7. Contient−sous−séquence−contiguë−non−fréquente = FAUX; // antimonotonicité
8. **Pour chaque** sous séquence de longueur $k-1$, $s' \subset s$ **faire**
9. **Si** $s' \notin F_{k-1}$ **alors** Contient−sous−séquence−contiguë−non−fréquente = VRAI;
10. **Si** ¬ Contient−sous−séquence−contiguë−non−fréquente **alors** ajouter *s* à C_k,;
11. **Fin;**
12. **Retourner** C_k..

Procédure : *Compter−supports*(*Ck*) − Calcul des supports des séquences candidates.

1. **Pour chaque** séquence de données $S \in D'$ **faire**
2. **Pour chaque** séquence candidate $c \in C_k$ **faire Si** $c \subseteq S$ **alors** Incrémenter Sup(*c*). // supportée

La fonction Gen−cand(Fk−1) est appelée à chaque itération k afin de générer l'ensemble des k−séquences candidates noté Ck, à partir de Fk−1, l'ensemble des (k−1)−séquences fréquentes, trouvé à l'itération précédente (k−1). La marche à suivre, consiste à réaliser une auto−jointure de Fk−1 sur lui-même.

Il s'agit de fusionner entre elles, deux à deux, toutes les séquences de Fk−1 afin de construire les différentes possibilités de séquences de longueur k. Une séquence S1 joindra une séquence S2 si la sous séquence obtenue en retirant le premier item de S1 est la même que celle obtenue en retirant le dernier item de S2.

La séquence générée, notée S1⊗ S2, n'est autre que S1 à laquelle a été ajouté à la fin le dernier item de S2.

Chaque nouvelle séquence générée est automatiquement, avant d'être insérée dans Ck, vérifiée qu'elle n'inclue aucune sous séquence contiguë non fréquente,), auquel cas elle serait non fréquente selon la propriété d'antimonotonicité. Ceci permet d'élaguer à priori toutes les séquences dont le calcul de leurs supports serrait assurément un traitement inutile. Quant aux séquences non élaguées, elles sont retournées dans Ck pour le calcul de leurs supports.

Le calcul des supports des séquences candidates est réalisé par Compter−supports(Ck), qui effectue un parcours complet de toute la base de séquences de données, en

incrémentant pour chaque séquence de données lue, les supports des séquences candidates de Ckq qui sont supportées par cette séquence, autrement dit, sous séquences de cette séquence.

Il ne reste pour cette k ème itération que de ne considérer dans Fk parmi les séquences de Ck, que celles ayant le support minimum définit par l'utilisateur (ligne 6). L'ensemble F de toutes les séquences fréquentes est ensuite mis à jour, et c'est cet ensemble qui est réduit en fin de recherche pour ne retourner en résultat que les séquences fréquentes maximales.

Application à un exemple

Cet exemple explique l'algorithme général d'extraction de motifs séquentiels APRIORI−ALL appliqué à la base de séquences de données D' représentée par le Tableau 2.5. Nous considérons un seuil de support minimum de 40% (soit 2/5).

CID	*SEQUENCES DE DONNEES*
C_1	<{A } {E}>
C_2	<{A} {B,C,D}>
C_3	<{A,C}>
C_4	<{A} {B,C,D} {E}>
C_5	<{E}>

Table 2.5 – Base de séquences de données exemple pour l'algorithme général d'extraction de motifs séquentiels

Itération (1) : Chaque item de I est considéré comme une 1−séquence candidate dans l'ensemble C1. Un premier parcours de D' permet alors de trouver leurs supports. Seules les 1−séquences fréquentes, c'est-à-dire, de support supérieur ou égal à 2/5 sont gardées dans F1.

C_1
1-séquences
<{A}>
<{B}>
<{C}>
<{D}>
<{E}>

parcours de D' →

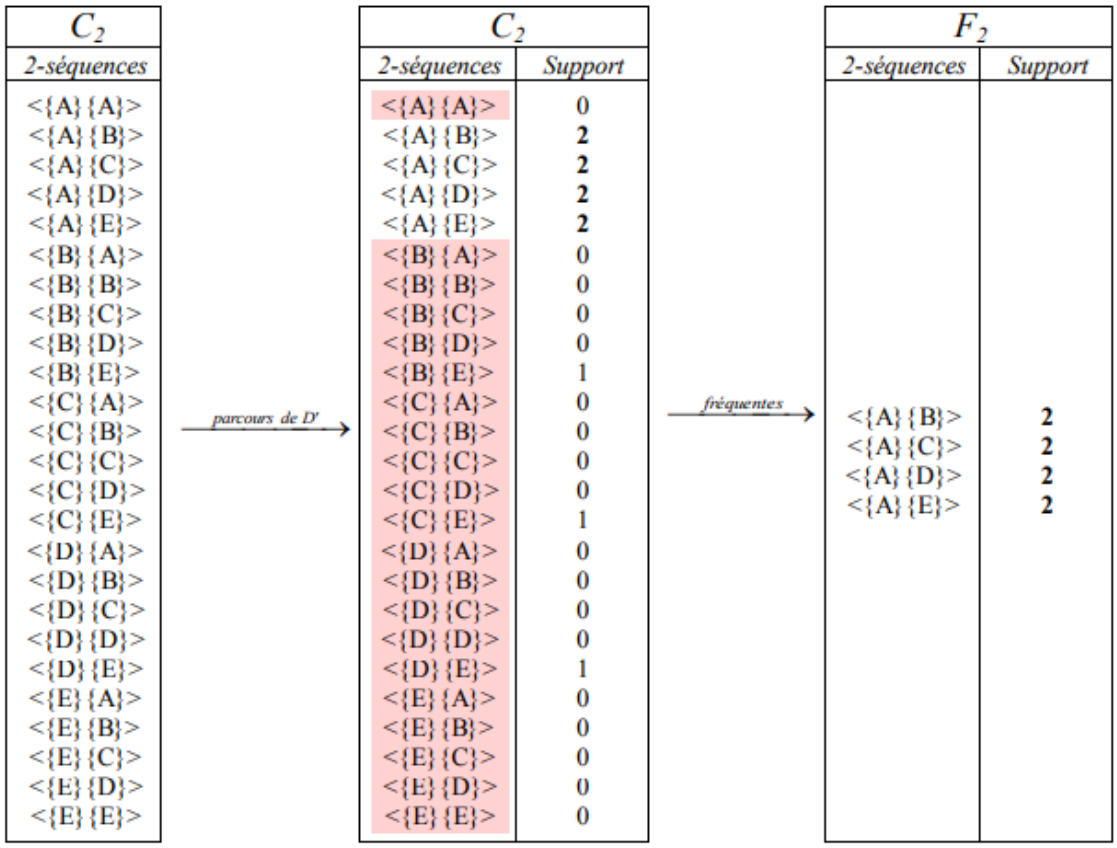
C_1	
1-équences	*Support*
<{A}>	**4**
<{B}>	**2**
<{C}>	**3**
<{D}>	**2**
<{E}>	**3**

fréquentes →

F_1	
1-séquences	*Support*
<{A}>	**4**
<{B}>	**2**
<{C}>	**3**
<{D}>	**2**
<{E}>	**3**

Itération (2) : APRIORI−ALL effectue une auto−jointure F1 ⊗F1 afin de générer C2,

L'ensemble des 2−séquences candidates. L'algorithme n'a pas besoin dans cette itération de s'assurer que chaque 2−séquence candidate générée ne contienne de sous séquences contiguës non fréquentes. Un second parcours de D' permet alors de calculer les supports de toutes les 2−séquences candidates. Seules les 2−séquences fréquentes sont gardées dans F2.



Itération (3) : L'algorithme arrête sa recherche puisque aucune séquence candidate ne peut , car on ne peut en aucun cas trouver la même séquence en∅ F2 = ⊗être générée à partir de F2 combinant deux à deux toutes les 2−séquences fréquentes de F2.

L'union F2YF1 constitue l'ensemble F ={<{A}>:4, <{B}>:2, <{C}>:3, <{D}>:3, <{E}>:3, <{A}{B}>:2, <{A}{C}>:2, <{A}{D}>:2, <{A}{E}>:2}de toutes les séquences fréquentes trouvées. Cet ensemble est ensuite réduit afin de ne retourner en résultat que les séquences fréquentes maximales (celles n'étant sous séquences d'aucune autre séquence fréquente), c'est-à-dire, l'ensemble FM = {<{A}{B}>:2, <{A}{C}>:2, <{A}{D}>:2, <{A}{E}>:2}.

Discussion

Il est important de mentionner que la phase de génération de séquences candidates est une tâche complexe qui peut être très coûteuse en temps UC et en espace mémoire dans le cas d'une base de données très dense ou d'un nombre d'items très élevé. C'est pourquoi, plusieurs alternatives vont être étudiées dans le prochain chapitre pour cette tâche.

Concernant l'étape de calcul des supports, rappelons qu'une séquence candidate n'est supportée par une séquence de données que si elle en est sous séquence tout en respectant l'ensemble des contraintes temporelles qui peuvent être spécifiées par l'utilisateur. Néanmoins, ces contraintes ne sont pas prises en charge dans APRIORI−ALL, comme nous l'avons constaté. C'est pourquoi, notre exemple s'est contenté de la contrainte de support minimum pour expliquer la démarche générale, afin de pouvoir aborder dans le prochain chapitre d'autres algorithmes plus complexes gérant ces paramètres.

Le dernier point important concerne le nombre de parcours coûteux de la base de données qui est égal à la longueur de la plus longue séquence fréquente trouvée (comme le montre clairement notre exemple), ce qui peut réduire considérablement les performances en temps de réponse de l'algorithme, notamment pour un seuil de support minimum trop faible. Nous reprendrons avec plus d'explications ces deux points au niveau du prochain chapitre.

6. Conclusion

Les motifs séquentiels, en plus de leur précision temporelle, présentent l'avantage d'être facilement interprétables. En effet, les règles qu'ils définissent montrent aisément comment des évènements se situent les uns par rapport aux autres dans le temps. Nous venons de présenter les fondements de base de cette technique. En partant du principe que les règles d'association en sont l'origine, nous avons fait le passage descriptif nécessaire. Ensuite, nous avons décrit quelques domaines intéressants dans lesquels les motifs séquentiels trouvent de larges champs d'application et obtiennent des résultats aisément exploitables.

Quant à la tâche d'extraction de ces motifs, il est utile de conclure ici que c'est une tâche non dirigée de Data mining. Toutefois, leur recherche peut considérer dans certains cas, en plus du seuil de support minimum spécifiée par l'utilisateur, une description à priori des motifs espérés, initiée par le même utilisateur qui introduit un ensemble de contraintes sous forme de paramètres temporels afin de la restreindre à seulement ce qui l'intéresse. Nous avons abordé ce concept en expliquant la notion de motifs séquentiels généralisés.

Ensuite, nous avons décrit le processus d'extraction des connaissances adapté à cette problématique et conclu que la recherche de séquences fréquentes constitue la phase la plus coûteuse de ce processus. Nous insistions également sur la phase de préparation des données, car de la qualité des données en entrée que dépend la qualité des résultats. Quant à la phase de visualisation des résultats, soulignons que le développement d'un mode de visualisation adapté à l'application concernée et aux attentes des utilisateurs, peut être grandement facilité à l'aide d'outils de visualisation de connaissances.

Pour la tâche d'extraction de séquences fréquentes, nous avons décrit la démarche générale relative, en se basant sur l'algorithme de base dans ce domaine, APRIORI−ALL [16]. Ce dernier, par sa recherche itérative par niveau en plus de sa non considération de contraintes temporelles, ne présente à notre connaissance, aucune minimisation du temps de réponse notamment, par ses nombreux parcours coûteux de la base de données. De plus, il souffre de limitations mémoire face au nombre phénoménal de séquences candidates qu'il peut générer pour un seuil de support minimum très faible, un nombre très élevé d'items ou dans le cas d'une base de données très dense au sens de ce support.

Ces facteurs seront sans doute, les éléments clés que nous considérerons dans le prochain chapitre notamment, lorsque nous savons maintenant que notre problème est fortement lié à un aspect combinatoire très complexe pouvant réduire considérablement, non seulement l'efficacité mais aussi l'utilité de la tâche d'extraction.

3
Approches d'extraction de motifs séquentiels

1 .Introduction

Elaborés en premiers lieux, les algorithmes de recherche de motifs fréquents connaissent de grandes difficultés d'adaptation au problème d'extraction de motifs séquentiels. En effet, si la découverte de ces motifs est à l'origine de celle des motifs séquentiels, les études dans ce sens montrent que, lorsque l'adaptation est possible, c'est au prix de temps de réponse inacceptables [18]. Depuis la définition du problème dans [16], de nombreuses approches visant à résoudre cette problématique ont été proposées, notamment avec l'objectif de minimiser le nombre de parcours de la base de données ainsi que celui de séquences candidates qui peuvent être générées en nombre important pendant la recherche.

Ce sont les deux facteurs essentiels de l'efficacité des algorithmes d'extraction de motifs séquentiels. L'importance du nombre de parcours de la base de données est principalement donnée par les énormes coûts d'opérations d'entrée/sortie que ces parcours engendrent.

L'importance du nombre de séquences candidates générées vient du fait que le traitement de ces dernières constitue pratiquement la majeure partie des temps de calcul et de l'espace mémoire que requièrent les types d'algorithmes générant ces séquences.

En se basant sur ces deux critères et d'autres que nous avons définis à cet effet, le présent chapitre étudie de plus près le problème d'extraction de motifs séquentiels dans une base de données. Il est organisé comme suit :

– La section 2 dresse un état de l'art des différents travaux de recherche dans ce domaine, tout en présentant les méthodes proposées, chacune dans le cadre de l'approche dont elle est issue. Nous complétons chaque description de méthode ou d'algorithme par une discussion où nous montrons plus particulièrement les apports et les limitations.

– La section 3 est consacrée à la présentation d'une synthèse des algorithmes présentés.

Nous comparons les performances ainsi que les structures de données utilisées, en s'appuyant notamment sur les temps de réponse et l'utilisation de l'espace mémoire. Une considération particulière est également donnée à la capacité de prise en charge de contraintes temporelles dans les motifs extraits par ces algorithmes.

Nous concluons ce chapitre par un bilan de points importants abordés tout en rapportant quelques conclusions communément tirées de l'étude comparative effectuée ainsi que des travaux que nous avons consulté.

2. Etat de l'art sur le problème du temps d'extraction de motifs séquentiels

Comme nous l'avons montré dans la Section 5 du chapitre précédent, l'algorithme de base APRIORI−ALL [16] effectue un nombre de parcours de la base de séquences de données égal à la longueur de la plus longue séquence fréquente trouvée. Partant du principe que ces parcours sont généralement répétés sur une mémoire secondaire (par le volume souvent important de la base de données), ceci peut réduire considérablement les performances en temps de réponse de l'algorithme, notamment lorsque ce dernier se trouve face à des données très denses au sens du seuil de support minimum spécifié par l'utilisateur et en extrait par conséquent, de plus en plus de séquences fréquentes plus longues.

D'autre part, cet algorithme se retrouve face à un très grand nombre de séquences candidates générées au cours de ses différentes phases pouvant augmenter de manière phénoménale dans le cas d'un très grand nombre d'items ou lorsque le seuil de support minimum spécifié par l'utilisateur est trop faible, ce qui peut constituer une grande limitation mémoire et réduire par conséquent, non seulement l'efficacité mais aussi l'utilité de la tâche.

Analyse du problème

Le problème d'extraction de motifs séquentiels dans une base de données est plus complexe que celui d'extraction d'itemsets fréquents. Il est évident que la recherche d'itemsets fréquents est seulement un cas particulier de la recherche de motifs séquentiels, en le considérant comme des séquences à un seul itemset. De plus, l'extraction de motifs séquentiels vise à coté de la découverte d'itemsets fréquents, l'arrangement de ces itemsets dans des séquences et l'identification de celles qui sont fréquentes. [19]

Pour comprendre pourquoi il existe une augmentation significative du nombre de motifs potentiels dans le cas des motifs séquentiels, considérons une base de données à

analyser avec n différents items. L'objectif de la recherche d'itemsets fréquents est de trouver tous les itemsets qui sont fréquents parmi ψ itemsets candidats, avec

$$\psi = \sum_{j=1}^{n} \binom{n}{j} - 1 = 2^n - 1$$

Pour comprendre le problème d'extraction de motifs séquentiels, commençons par considérer que la base de données contient des séquences avec au plus *m* itemsets et chaque itemset a au plus un item (1−itemset). Dans ces conditions, on aura *n*×*m* différentes séquences possibles avec *m* itemsets et différentes séquences possibles de longueur arbitraire.

$$\sum_{k=1}^{m} n^k = \frac{n^{m+1} - n}{n - 1}$$

De même, si chaque itemset a un nombre arbitraire d'items, il existerait ψ_m séquences fréquentes possibles avec $\psi_m = \psi^m = (2^n\text{-}1)^m$et, il y aurait $\psi_s$séquences candidates en Général, avec

$$\psi_s = \sum_{k=1}^{m} (2^n - 1)^k = \frac{(2^n - 1)^{m+1} - 2^n - 1}{2^n - 2} = \Theta(2^{nm}) \qquad (2)$$

L'équation (2) montre la taille énorme de l'espace de recherche à explorer pour l'extraction de motifs séquentiels dans une base de données. Cette complexité spatio−temporelle nécessite une approche efficace qui, étant donné le grand écart entre le nombre de séquences candidates et celui des séquences fréquentes, doit déterminer quelles séquences candidates considérer, puis trouver efficacement lesquelles parmi celles-ci sont fréquentes.

C'est pourquoi, notamment dans l'élagage efficace de l'espace de recherche, les méthodes de projection de la base de données [20], les techniques d'indexation [21], l'extraction sans génération de séquences candidates [22] et l'exploitation de structures de données adaptées [23].

Nous proposons dans cette section de faire un pas en avant dans la compréhension de ces techniques en expliquant les lignes directrices des grands algorithmes les implémentant et en

discutant leurs performances notamment en temps de traitement et en espace mémoire tout en évoquant la composante qui nous intéresse à savoir : l'extraction de motifs séquentiels sous contraintes temporelles (motifs séquentiels généralisés).

Stratégie de résolution Etant donnée une base de données temporelle *D* et un ensemble d'items I. Soit *D'* la transformée de *D* en base de séquences de données et *MinSup* le seuil de support minimum spécifié par l'utilisateur. Considérons [*MinGap, MaxGap*] et *WinSize*, les contraintes temporelles (intervalle de temps et fenêtre temporelle) qui peuvent être spécifiées par l'utilisateur en plus du seuil de support minimum.

Le problème à résoudre consiste alors à trouver l'ensemble de toute la séquence fréquente maximales dans la base de données *D*, ce qui revient à extraire à partir de sa transformée en base de séquences de données *D'*, l'ensemble de toutes les séquences ayant un support supérieur ou égal à *MinSup*, autrement dit, contenues au moins *MinSup* fois dans *D'* tout en respectant les contraintes temporelles {*MinGap, MaxGap, WinSize*} dans le cas où elles sont spécifiées. Et c'est cet ensemble qui est finalement réduit afin de ne retourner à l'utilisateur que les séquences fréquences maximales. [24]

Dans ce qui suit, cette dernière tâche ne sera pas détaillée car c'est une tâche particulièrement simple (algorithme de détection d'inclusions), souvent commune à toutes les approches d'extraction de motifs séquentiels. Notre problème se restreindra alors à l'extraction de l'ensemble des séquences fréquentes.

Les méthodes existantes différentes essentiellement sur la manière de parcourir l'espace de recherche (largeur d'abord ou profondeur d'abord) et sur les structures de données utilisées pour sous forme de taxonomie indexer la base de données et faciliter une énumération rapide. Les algorithmes d'extraction de motifs séquentiels peuvent être classés en trois grandes catégories :

- ***Méthodes horizontales***
- ***Méthodes verticales***
- ***Méthodes par projection***

Comme le montre la figure suivante :

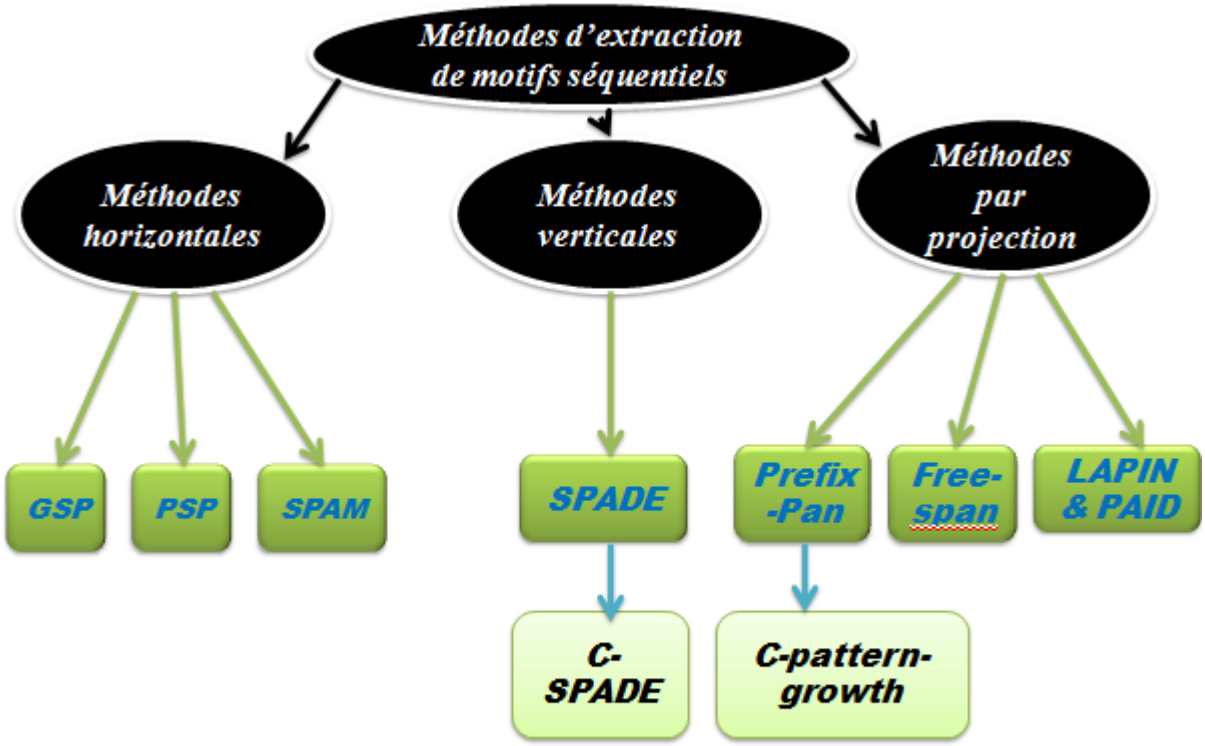


Figure 3.1– Méthodes d'extraction de motifs séquentiels

Nous décrivons par la suite ces trois catégories ainsi que les principaux algorithmes associés.

2.1. Méthodes horizontales

2.1.1. Algorithme pionnier GSP et sa structure

Comme nous l'avons mentionné dans le chapitre précédent, la première proposition pour découvrir les motifs séquentiels était l'algorithme de base ***APRIORI−ALL***[16]. Cependant, aucune contrainte temporelle ne pouvait être spécifiée à l'aide de cet algorithme. Afin de corriger cette insuffisance, les mêmes auteurs ont proposé d'y intégrer la possibilité de définir par l'utilisateur d'un intervalle de temps ainsi que d'une fenêtre temporelle, ce qui a donné naissance à l'algorithme pionnier en recherche de motifs séquentiels généralisés : l'algorithme **GSP (Generalized Sequential Patterns)** [17].

2.1.1.1 principes de base

GSP reprend le principe d'une recherche itérative *par niveau* de l'algorithme *APRIORI−ALL*.

Pour ceci, il débute par un tri de la base de données initiale en se basant sur l'identifiant unique *CID* comme clé primaire et comme clé secondaire, l'identifiant temporel de cette base, afin de la transformer en une base de séquences de données, et c'est cette dernière qui est analysée par l'algorithme, comme le montre l'exemple suivant.

Exemple

La Figure 3.2 représente une base de données temporelle D qui sera triée en utilisant comme clé primaire l'identifiant unique du client CID et comme clé secondaire la date de transaction, pour être aisément transformée en une base de séquences de données D'.

CID	*DATE*	*ITEMSET*
C_2	01/01/2008	{B}
C_3	01/01/2008	{A}
C_2	02/01/2008	{A}
C_1	02/01/2008	{B}
C_3	03/01/2008	{A}
C_1	03/01/2008	{A}
C_2	04/01/2008	{C}
C_1	05/01/2008	{C}
C_2	06/01/2008	{B}
C_3	07/01/2008	{C}
C_1	07/01/2008	{B}

Tri →

CID	*DATE*	*ITEMSET*
C_1	02/01/2008	{B}
	03/01/2008	{A}
	05/01/2008	{C}
	07/01/2008	{B}
C_2	01/01/2008	{B}
	02/01/2008	{A}
	04/01/2008	{C}
	06/01/2008	{B}
C_3	01/01/2008	{A}
	03/01/2008	{A}
	07/01/2008	{C}

Transformation ⇓

D'	01/01/08	02/01/08	03/01/08	04/01/08	05/01/08	06/01/08	07/01/08
C_1		{B}	{A}		{C}		{B}
C_2	{B}	{A}		{C}		{B}	
C_3	{A}		{A}				{C}

Figure 3.2 – Transformation GSP d'une base de données temporelle en une base de séquence de données.

À la première itération, *GSP* effectue un premier parcours de la base de séquences de données *D'afin* de calculer le support de chaque item, autrement dit, le nombre de séquences de données (séquences clients) dans *D'* contenant au moins une fois cet item, ce qui permet de déterminer *F*1, l'ensemble des 1−séquences fréquentes. L'algorithme alterne ensuite entre phases de génération de séquences candidates, c'est-à-dire, celles dont on ne connaît pas encore le support, et phases de calcul des supports des

séquences générées afin de déterminer les fréquentes parmi celles-ci. On notera par *Ck* l'ensemble des *k*−séquences candidates à, l'itération *k* et par *Fk* l'ensemble des *k*−séquences fréquentes à l'itération *k*.

De manière générale, l'ensemble *Fk*−1 des (*k*−1)−séquences fréquentes de l'itération (*k*−1) est utilisé pour générer l'ensemble *Ck* des *k*−séquences candidates à l'itération *k*. La technique par laquelle *GSP* opère pour effectuer cette tâche sera expliquée dans la ***Section 2.1.1.2.*** Une fois l'ensemble *Ck* généré, la base de séquences de données *D'* est parcourue afin de calculer le support de chaque séquence candidate générée. La manière par laquelle *GSP* opère pour vérifier le support d'une séquence candidate par une séquence de données sera expliquée dans la ***Section 2.1.1.3***. Parmi les séquences candidates de *Ck*, seules les séquences fréquentes, c'est-à-dire, celles ayant le seuil de support minimum, constitueront l'ensemble *Fk* des séquences fréquentes de l'itération suivante k.

GSP arrête la recherche lorsque aucune autre séquence candidate ne peut être générée ou lorsque aucune autre séquence fréquente ne peut être identifiée. L'ensemble *F* regroupant toutes les séquences fréquentes trouvées, représentant l'ensemble des motifs séquentiels découverts par cet algorithme est donc retourné en résultat.

2.1.1.2. Génération de séquences candidates

La phase de génération de *Ck*, l'ensemble des *k*−séquences candidates de l'itération *k* à partir de *Fk*−1, l'ensemble des (*k*−1)−séquences fréquentes de l'itération (*k*−1), se fait en réalisant une *jointure* de *Fk*−1 avec lui-même, appelée *auto−jointure* de *Fk*−1. La technique utilisée par *GSP* pour réaliser cette tâche est essentiellement basée sur la propriété suivante.

Propriété 01 (jointure de deux séquences)

Une séquence S1 joindra une séquence S2 si la sous séquence obtenue en retirant le premier item de S1 est la même que celle obtenue en retirant le dernier item de S2. La séquence générée notée S = S1 ⊗ S2 n'est autre que S1 à laquelle a été ajouté à la fin le dernier item de S2 qui devient séparé dans S s'il l'était dans S2, sinon, il est ajouté au dernier itemset de S1.

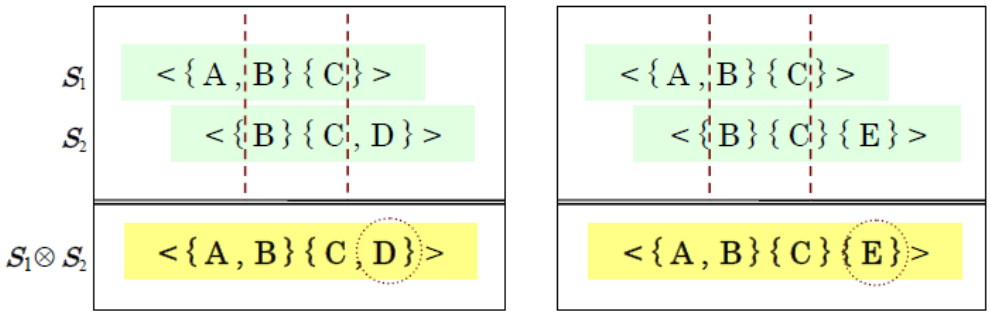


Figure 3.3 – jointure de séquences dans GSP

Exemples de jointures

- <{A,B}{C}> ⊗ <{B}{C,D}> ⟶ <{A,B}{C,D}>
- <{A,B}{C}> ⊗ <{B}{C}{E}> ⟶ <{A,B}{C}{E}>
- <{A}{C}{D}> ⊗ <{C}{D,E}> ⟶ <{A}{C}{D,E}>
- <{A}{A}{A}> ⊗ <{A}{A}{A}> ⟶ <{A}{A}{A}{A}>
- <{A}> ⊗ <{B}> ⟶ <{A}{B}>, <{B}{A}>, <{A,B}>

Le théorème suivant est très important puisqu'il garantit que pour toutes les longueurs de séquences fréquentes trouvées, si une séquence candidate est susceptible d'être fréquente alors elle sera générée par l'auto−jointure ci-dessus décrite.

Théorème 01 :

Soit Fk l'ensemble des k−séquences fréquentes, la génération de séquences candidates par la Jointure de Fk sur lui-même construit un sur-ensemble de Fk+1.

Le dernier point important concerne l'optimisation de l'espace mémoire occupé par le séquences candidates générées ainsi que des coûts de calcul de leurs supports : *GSP* s'assure pour chaque nouvelle séquence candidate générée qu'elle ne contienne aucune sous séquence contiguë non fréquente, auquel cas elle serait elle-même non fréquente selon la propriété d'antimonotonicité. Elle sera donc élaguée de l'ensemble des séquences candidates.

2.1.1.3. Calcul des supports

Il s'agit pour *GSP* d'effectuer un **parcours complet** de toute la base de séquences de données afin d'incrémenter pour chaque séquence de données lue, les supports des

séquences candidates dans *Ck* qui sont supportées par cette séquence. Rappelons qu'une séquence candidate n'est supportée par une séquence de données que si elle en est sa sous séquence.

Toutefois, dans le cas où des paramètres temporels (intervalle de temps et fenêtre temporelle) sont spécifiés par l'utilisateur en plus du seuil de support minimum (Il existe même des méthodes automatiques permettant d'optimiser ces paramètres, l'algorithme doit s'assurer que la séquence candidate respecte ces contraintes avant d'en augmenter le support. Cette vérification se fait dans *GSP* à l'aide d'une procédure qui alterne entre deux phases de recherche : la *phase AVANT* et la *phase ARRIÈRE*.

La phase avant

Considérons une séquence candidate quelconque *S*1 à vérifier le support par une séquence de données *S*. Il s'agit pour *GSP* de la lecture successive de chaque itemset (transaction) de *S* afin d'y trouver une sous séquence *S'* contenant au moins la séquence candidate *S*1 et respectant l'ensemble {*MinGap, MaxGap, WinSize*} des contraintes imposées par les paramètres temporels. Dans le cas où à un moment donné durant la recherche, une ou plusieurs de ces contraintes sont violées, l'algorithme passe à la *phase arrière*.

La phase arrière

Lorsque la distance temporelle (définie exclusivement à la base de l'identifiant temporel, voir *Section 3* du *Chapitre 2*) écoulée entre le dernier itemset *Si* découvert durant la *phase avant* et l'itemset *Si* −1 découvert précédemment, ne respecte pas l'une des contraintes temporelles, l'algorithme tente de trouver un nouveau itemset dans *S* qui contient *Si*−1 tout en respectant ces contraintes. La phase arrière se poursuit aussi longtemps que tous les itemsets de *S* ne respectent pas ces contraintes. Si cette recherche réussit, *GSP* retourne à la *phase avant*, sinon, la séquence candidate examinée est non supportée par la séquence de données. L'algorithme tente de trouver un nouveau itemset dans *S* qui contient *Si*−1 tout en respectant ces contraintes. La phase arrière se poursuit aussi longtemps que tous les itemsets de *S* ne respectent pas ces contraintes. Si cette recherche réussit, *GSP* retourne à la *phase avant*, sinon, la séquence candidate examinée est non supportée par la séquence de données.[25]

Application à un exemple

Le présent exemple explique l'algorithme *GSP*, appliqué à la base de séquences de données *D'* de la *Figure 3.2* avec un seuil de support minimum de 2/3. L'intervalle de

temps est placé à [*MinGap*= 0, *MaxGap*= 3] jours et la fenêtre temporelle à *WinSize*= 2 jours.

Itération (1) : Chaque item de la base *D'* est une 1−séquence candidate dans *C1*. Un premier Parcours de *D'* permet de calculer les supports de toutes ces 1−séquence candidates. Parmi Celles-ci, seules les 1−séquences fréquentes sont gardées dans *F1*.

C_1
1−séquences
<{A}>
<{B}>
<{C}>

$\xrightarrow{D'}$

C_1	
1−séquences	*Supports*
<{A}>	**3**
<{B}>	**2**
<{C}>	**3**

$\xrightarrow{f}$

F_1	
1−séquences	*Supports*
<{A}>	**3**
<{B}>	**2**
<{C}>	**3**

Itération (2) : *GSP* effectue l'auto−jointure *F1* ⊗*F1*afin de générer *C2* l'ensemble des 2−séquences candidates : toutes les 1−séquences de *F1* sont fusionnées entre elles puisque dans tous les cas l'action de retirer le premier et le dernier item donne la même séquence vide. Un second parcours de *D'* permet de calculer les supports de chacune des 2−séquences candidates générées pour ne garder dans *F2* que les 2−séquences fréquentes.

C_2
2−séquences
<{A}{A}>
<{A}{B}>
<{A}{C}>
<{B}{A}>
<{B}{B}>
<{B}{C}>
<{C}{A}>
<{C}{B}>
<{C}{C}>
<{A,B}>
<{A,C}>
<{B,C}>

$\xrightarrow{D'}$

C_2	
2−séquences	*Supports*
<{A}{A}>	1
<{A}{B}>	0
<{A}{C}>	**2**
<{B}{A}>	0
<{B}{B}>	0
<{B}{C}>	**2**
<{C}{A}>	0
<{C}{B}>	**2**
<{C}{C}>	0
<{A,B}>	**2**
<{A,C}>	0
<{B,C}>	0

$\xrightarrow{f}$

F_2	
2−séquences	*Supports*
<{A}{C}>	**2**
<{B}{C}>	**2**
<{C}{B}>	**2**
<{A,B}>	**2**

Afin de montrer l'effet des paramètres temporels sur le calcul des supports, détaillons les calculs des supports des deux séquences candidates <{A}{C}> et <{A,B}> :

Pour la séquence candidate <{A}{C}

L'algorithme débute par la phase *avant*, parcourt la séquence de données *C1* et identifie l'item A au temps 3. Il poursuit son avancement et identifie C au temps 5. Puisque 5−3 ≤ *MaxGap*, alors la séquence *C1* supporte cette séquence et son support est donc incrémenté. De même pour *C2*. Pour *C3*, l'algorithme cherche l'item A et l'identifie au

temps 1, puis l'item C et l'identifie au temps 7. Puisque 7−1 >*MaxGap*, il passe à la phase *arrière* et retire A au temps 1 puis part à sa recherche à partir du temps 4 = 7 – *MaxGap*. *GSP* arrive à la fin de *C3* sans avoir identifié un nouvel item A. Par conséquent, C3 ne supporte pas <{A}{C}>. Une illustration schématisée de ce parcours est montrée dans la Figure 3.4 ci-dessous.

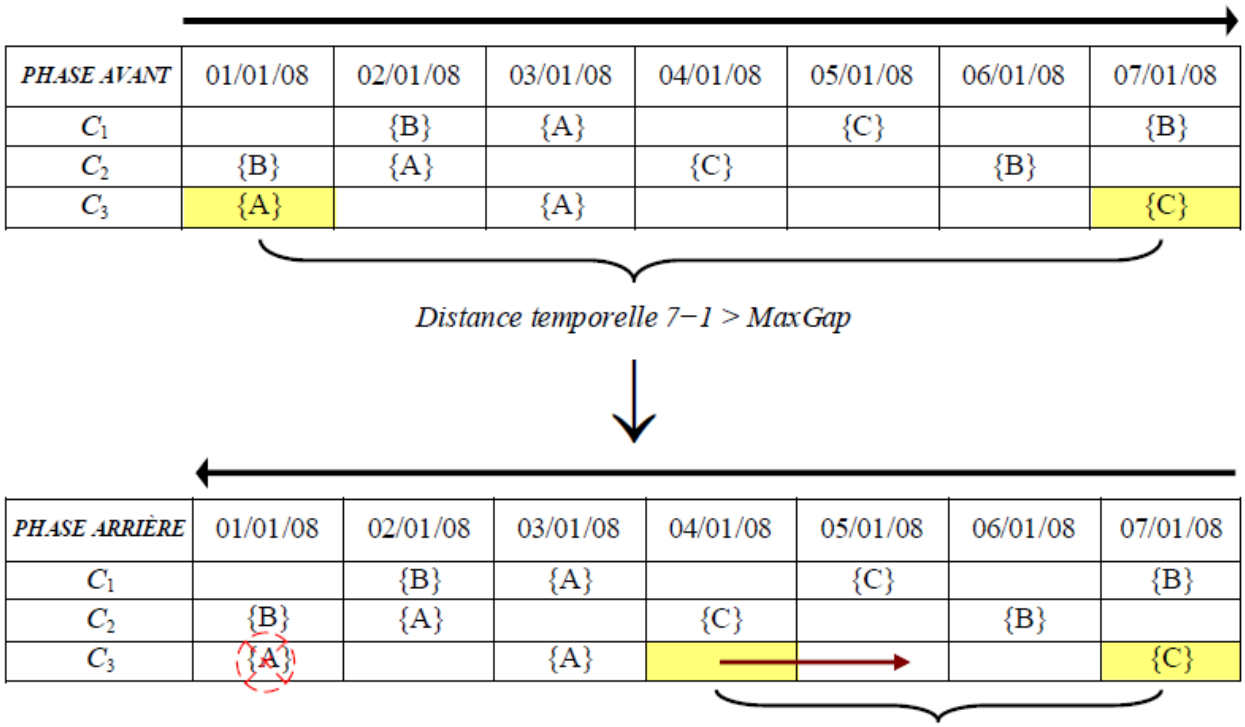


Figure 3.4–Vérification GSP du support de la séquence candidate < {A}{C}> par la séquence de données C*3* avec *MaxGap*= 3

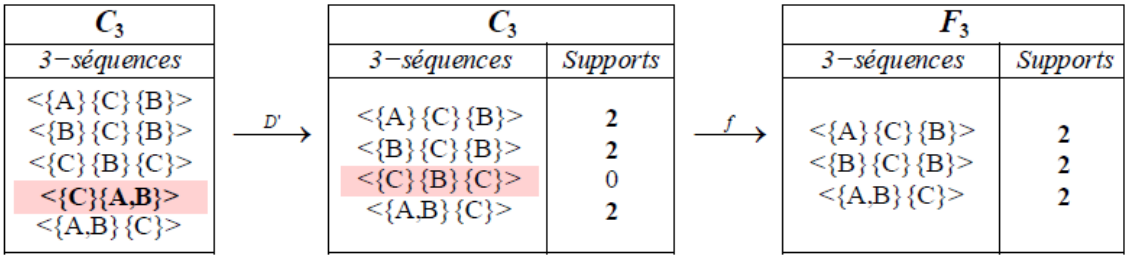
Pour la séquence candidate < {A,B}> :

L'algorithme débute par le parcours de la séquence C1 et identifie l'item B au temps 2, puis l'item A au temps 3. Comme 3−2 <WinSize, la séquence C1 supporte cette séquence. La séquence C2 supporte également cette séquence. Pour C3, l'algorithme identifie A au temps 1, puis parcourt le reste de la séquence C3 sans identifier l'item B. Par conséquent, la séquence C3 ne supporte pas la séquence < {A,B}>.

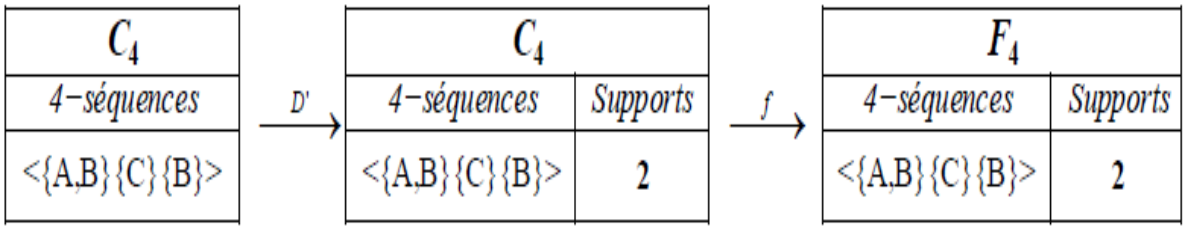
Itération (3) : Les 3−séquences candidates sont générées dans C3 par l'auto−jointure F2 ⊗F2.

Remarquons que <{B}{C}> et <{A,B}> n'ont pas pu être fusionnées puisque l'action de retirer le premier item de <{B}{C}> ne donne pas le même résultat que de retirer un item de <{A,B}> qui donne <{A}> ou <{B}>. GSP s'assure que les 3−séquences générées n'ont pas de 2−sous séquences contiguës non fréquentes, auquel cas, elles

seraient elles mêmes non fréquentes, comme la 3−séquence <{C}{A,B}> qui est élaguée puisque sa 2−sous séquence <{C}{A}> ne figure pas dans F2. Un 3ème parcours de D' permet de calculer les supports des 3−séquences candidates pour ne garder dans F3 que les 3−séquences fréquentes.



Itération (4) : L'auto jointure*F3⊗F3* permet de générer la seule 4−séquence candidate<{A,B}{C}{B}>, qui n'est pas élaguée car toutes ses 3−sous séquences contiguës sont dans *F3*. Un 4ème parcours de *D'* permet de calculer son support de 2. Elle est donc gardée dans *F4*.



Itération (5) : L'auto−jointure *F4⊗F4* = ∅, car la seule 4−séquence <{A,B}{C}{B}>dans *F4* ne peut être fusionnée avec elle-même. Par conséquent, aucune 5−séquence candidate ne peut être générée. L'algorithme arrête la recherche et retourne l'ensemble de toutes les séquences fréquentes trouvées, c'est-à-dire, $F = F_1 \cup F_2 \cup F_3 \cup F_4$qui représente les 11 motifs découverts : <{A}>:3, <{B}>:2, <{C}>:3, <{A}{C}>:2, <{B}{C}>:2, <{C}{B}>:2, <{A,B}>:2, <{A}{C}{B}>:2, <{B}{C}{B}>:2, <{A,B}{C}>:2 et <{A,B}{C}{B}>:2.

2.1.1.4 Représentation en arbre de hachage des séquences candidates.

GSP exploite une structure de données en ***arbre de hachage*** destinée à contenir la collection des séquences candidates générées. Chaque nœud intérieur de l'arbre est une table de hachage dont chaque entrée pointe vers un noeud de niveau immédiatement supérieur. Chaque chemin de l'arbre correspond donc à une succession de clefs de hachage. Quant aux feuilles de l'arbre, elles contiennent les séquences candidates qui

sont réparties dans ces feuilles en fonction de leurs préfixes. Un compteur de support est associé à chaque séquence.[26]

Exemple :

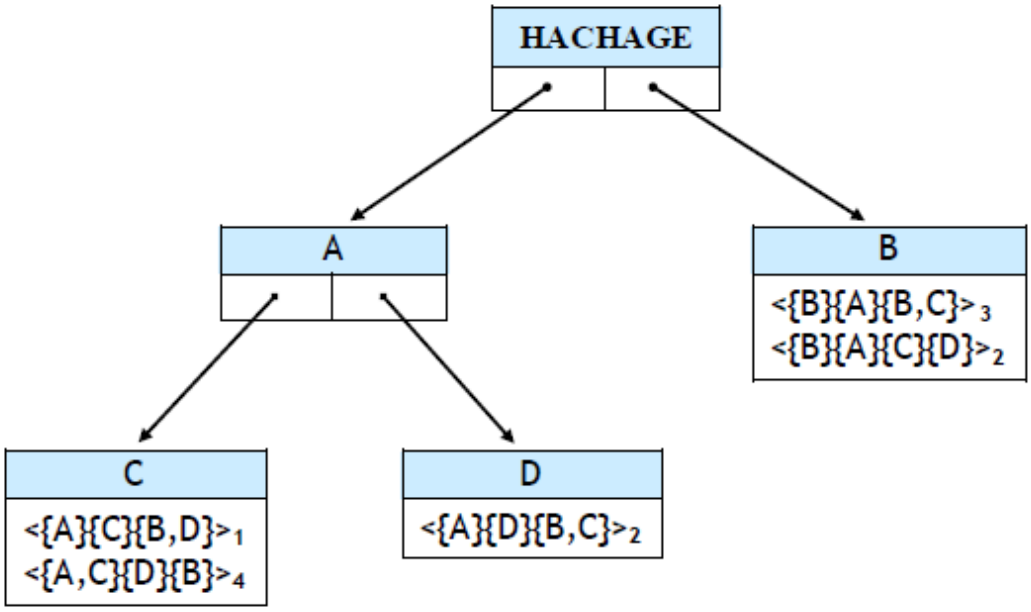


Figure 3.5 : Arbre de hachage GSP correspondant au niveau 4

La *Figure 3.5* montre un arbre de hachage *GSP* de niveau 4 représentant les4−séquences candidates : <{A}{C}{B,D}>, <{A,C}{D}{B}>, <{A}{D}{B,C}>, <{B}{A}{B,C}> et <{B}{A}{C}{D}> avec leurs supports respectifs : 1, 4, 2, 3 et 2. Une limite spécifiée à l'avance de 2 séquences maximum par feuille est considérée.

Chaque nouvelle séquence candidate générée par *GSP* pendant sa phase de génération de séquences candidates, est automatiquement insérée dans cet arbre. Pour ceci, l'algorithme parcourt l'arbre, en appliquant à chaque niveau *k* une fonction de hachage sur le *kème* item de la séquence, et ce de façon à atteindre la feuille dans laquelle il peut insérer cette séquence. Si l'ajout dans cette feuille de la séquence candidate porte le nombre de séquences contenues dans la feuille à un niveau supérieur à une limite spécifiée, alors la feuille est scindée en deux ou plusieurs feuilles, reprenant l'intégralité des séquences contenues dans la feuille mère, qui est transformée en un noeud intérieur pointant sur les feuilles nouvellement générées.

Lors du comptage des supports des séquences candidates, toutes les séquences de la base de séquences de données sont *projetées* dans cet arbre, en appliquant à chaque projection d'une séquence de données, la fonction de hachage sur les items de la dite séquence. Quand une feuille est atteinte, elle contient les séquences candidates

potentielles pour cette séquence de données. Il ne reste pour *GSP* que d'identifier parmi celles-ci, en faisant appel à son code de vérification de support (*phase Avant* et *phase Arrière*, voir ***Section 2.1.1.3*),** celles qui sont supportées par la séquence de données pour en incrémenter les supports.

2.1.1.5 Discussion

L'algorithme *GSP*, outre le fait qu'il est le premier à poser la problématique des motifs séquentiels généralisés a eu l'avantage de montrer qu'il était nécessaire d'implémenter une structure de données efficace permettant de mieux représenter les séquences candidates tout en optimisant le comptage de leurs supports. La structure en arbre de hachage apportée par cet algorithme est performante dans le sens où les séquences potentielles sont identifiées rapidement car le nombre de comparaisons par symbole entre séquences est minimal.

Toutefois, elle présente une faiblesse majeure. En effet, elle consiste à mettre en commun des préfixes sans faire de distinction entre deux items du même itemset et deux items avec changement d'itemset. La *Figure 3.5* montre clairement ce défaut. La feuille C est utilisée pour stocker les séquences candidates <{A}{C}{B,D}> et <{A,C}{D}{B}>. Quand *GSP* atteint cette feuille, il n'a aucun moyen de savoir si c'est la séquence <{A}{C}> ou bien <{A,C}> qui l'a conduit jusqu'à cette feuille. Tout l'inconvénient réside dans le fait que, même en l'absence de paramètres temporels, l'algorithme se voit contraint de vérifier à nouveau l'inclusion des séquences candidates contenues dans les feuilles atteintes, dans la séquence de données projetée, afin de savoir quels supports incrémenter ! La deuxième faiblesse de *GSP* réside dans son mode de recherche, basé sur le principe ***Tester−et Générer*** mis en place par l'algorithme *APRIORI*. Seulement pour *GSP*, le nombre de candidats est **énorme** par rapport à celui dans *APRIORI*. Ceci est imposé par le concept des motifs que cherche *GSP* : l'ordre des items lui est fondamental et il lui est par conséquent vital de devoir considérer dans différents ordres toutes les séquences possibles d'items ou de groupes d'items. Considérons par exemple *n* items trouvés fréquents lors de la première itération, *GSP* doit alors s'attendre à devoir générer, uniquement pour la seconde itération, pas moins de $n^2 + \frac{n(n-1)}{2}$

Séquences candidates de longueur 2 [28]. Cette complexité spatio−temporelle affectera sans doute les performances de l'algorithme, notamment en temps de réponse et en utilisation de l'espace mémoire. Le dernier point important concerne le nombre de parcours coûteux de la base de données.

GSP qui suit une recherche par niveau est contraint à chaque itération, d'en effectuer un parcours complet afin de calculer les supports des séquences candidates. Si **µ** est la longueur de la plus longue séquence fréquente trouvée, alors il aura effectué **µ** lectures de cette base.

Pour un **µ** très grand, les performances en temps de réponse peuvent nettement se dégrader. Ce cas se présente, notamment, pour une base de données **très dense** ou lorsque les contraintes temporelles sur les motifs sont **absentes** ou moins restrictives.[25]

2.1.2 Algorithme PSP

Comme expliqué dans la discussion précédente, l'arbre de hachage GSP consiste à mettre en commun des préfixes, sans faire de distinction entre deux items du même itemset et deux items avec changement d'itemset. De ce fait, l'algorithme se voit contraint lorsqu' il atteint les séquences candidates dans les feuilles, de vérifier à nouveau leur inclusion dans la séquence projetée dans l'arbre afin d'identifier celles qui sont réellement incluses dans cette séquence.

Comme solution à cette faiblesse majeure, [27] ont proposé un nouvel algorithme, appelé *PSP* **(*Prefix tree for Sequential Patterns*),** basé comme *GSP* sur une recherche par niveau, mais qui utilise une organisation différente des séquences candidates afin d'identifier plus rapidement celles qui sont supportées par une séquence de données.

2.1.2.1 Principe de base

Il s'agit de factoriser les séquences candidates selon leurs préfixes tout en considérant les changements d'itemsets dans cette factorisation. Pour cela, PSP propose une structure de données en arbre de préfixes.

Toute la différence avec l'arbre de hachage de GSP, réside dans le fait que les séquences candidates ne sont plus stockées dans les feuilles, mais représentées par l'arbre lui-même : Tout chemin de la racine vers un noeud de l'arbre représente une seule séquence candidate toute séquence candidate est représentée par un unique chemin de la racine vers un nœud.

Pour prendre en compte les changements d'itemsets, l'arbre est doté de deux types de branches entre ses nœuds. Le premier **"Same transaction**", signifie que les items sont dans le même itemset, alors que le second **"Other transaction"** signifie qu'il y a un changement d'itemset entre les deux items, comme le montre la Figure 3.6 suivante.

Exemple

La Figure 3.6 montre un arbre de préfixes PSP représentant les séquences candidates de l'arbre de hachage GSP de la Figure 3.5 , Elle montre que pour des séquences de longueur k, la profondeur de l'arbre est exactement k. Nous pouvons distinguer les items appartenant à un même itemset (branche en pointillés) ainsi que les changements d'itemsets (en trait plein).

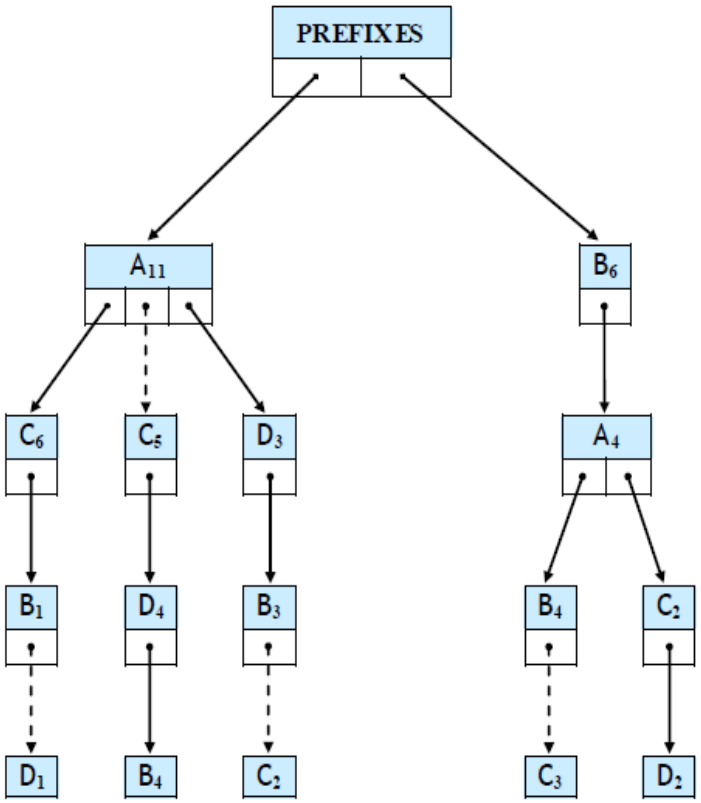


Figure 3.6 : Arbre de préfixes PSP correspondant au niveau 4

D'autres particularités de cette structure permettent de la distinguer de l'arbre de hachage GSP et plus particulièrement le fait que les séquences candidates et les séquences fréquentes sont toutes représentées par cette même structure. De plus, les traitements de PSP sont tous définis autour de cet arbre qui est géré de la manière suivante :

Au premier niveau (k=1) : Chaque item de la base de données est représenté par une feuille reliée à la racine par une branche de type Other transaction afin de représenter toutes les 1−séquences candidates. Chacune des feuilles contient alors l'item et son support. Un parcours complet de toute la base de séquences de données est nécessaire afin de calculer leurs supports. Ensuite, sont élaguées les feuilles dont les supports sont

inférieurs au seuil de support minimum afin de ne garder dans l'arbre que les 1−séquences fréquentes.

Pour les niveaux supérieurs (k≥2) : La génération des k−séquences candidates à partir des (k−1)−séquences fréquentes est particulièrement basée dans PSP sur son arbre de préfixes. La manière par laquelle cet algorithme opère, sera expliquée dans la Section **II.1.2.2** suivante.

Quant à la représentation des séquences générées, elle est réalisée de la manière suivante Chaque nouvelle k−séquence candidate S générée est automatiquement représentée dans l'arbre par un chemin de la racine vers une feuille nouvellement créée pour le dernier item S[k] de cette séquence. Pour les items S[1] à S[k−1], l'algorithme s'assure de rechercher dans l'arbre, l'existence d'un chemin pouvant représenter la plus longue sous séquence possible de S : < S[1]…S[n] > avec n ϵ [1, k−1], auquel cas la seconde <S[n+1]…S[k] > sera liée à celle-ci dans l'arbre. Cette factorisation permet de ne représenter qu'une unique fois, les préfixes communs à plusieurs séquences candidates. Un parcours de la base de séquences de données permet de calculer tous les supports, pour enfin, élaguer les feuilles dont les supports sont insuffisants et ne garder dans l'arbre, que les k−séquences fréquentes.

2.1.2.2 Génération de séquences candidates

La phase de génération de séquences candidates de longueur 2 est similaire à celle dans GSP.

Par contre, pour les niveaux supérieurs, PSP tire particulièrement profit de sa structure d'arbre de préfixes. Ceci est réalisé de la manière suivante : Pour chacune des feuilles l de l'arbre, PSP recherche à la racine l'item x représenté par l. Ensuite il étend la feuille l en construisant pour cette feuille une copie des fils de x. A cette étape, il applique un filtrage destiné à ne pas générer de séquences dont il sait à l'avance qu'elles ne peuvent être fréquentes. Pour cela, il considère F, l'ensemble des fils de x. Pour chaque f dans F, si f n'est pas frère de l alors il est inutile d'étendre l avec f. En effet, nous savons que si f n'est pas frère de l, alors soit p le père de l, on aura <p;f> n'est pas fréquente et donc <p;l;f> ne peut l'être.

Exemple

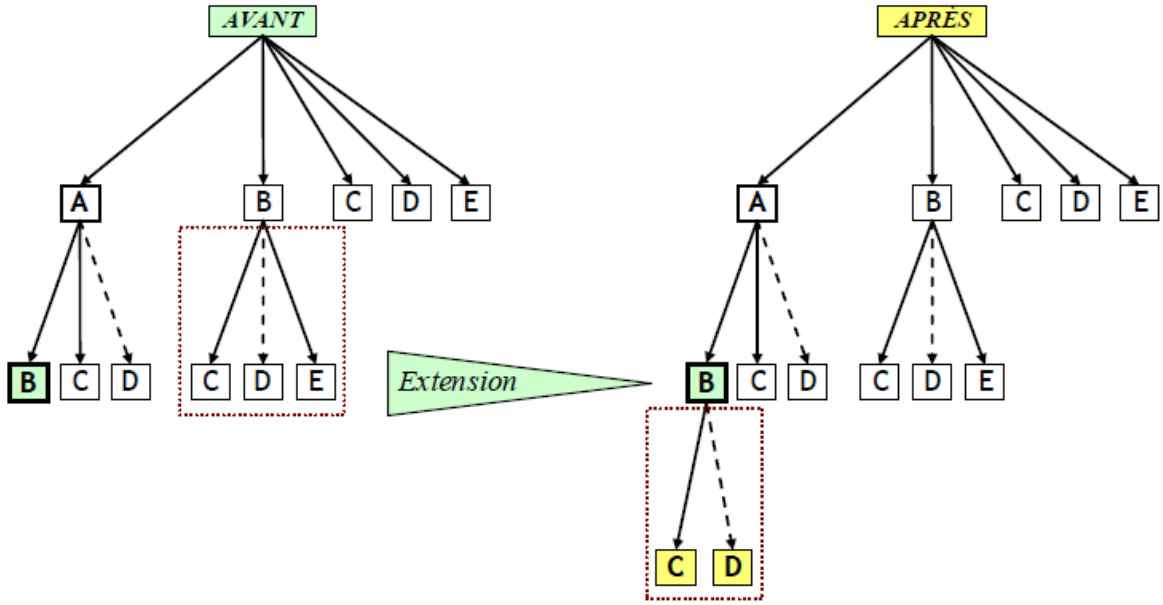


Figure 3.7 – Génération de séquences candidates dans PSP

La Figure 3.7 représente un arbre de préfixes PSP, Avant et Après la génération de 3−séquences candidates à partir de l'item B (en gras). Ce dernier est étendu dans l'arbre Après, uniquement avec ses fils de la racine qui sont ses frères dans sa branche de A, à savoir, l'item C **(Other transaction)** et l'item D **(Same transaction)**. L'item E est élagué puisqu'il n'est pas frère de B dans cette branche, autrement dit, les deux séquences <{A}{E}> et <{A,E}> ne sont pas dans l'arbre Avant, et donc sont non fréquentes. Ainsi, < {A}{B}{E}> et <{A}{B,E}> ne peuvent l'être selon la propriété d'antimonotonicité, et il est donc inutile de générer ces deux séquences non fréquentes. En somme, PSP aura généré à partir du noeud B, uniquement les deux 3−séquences candidates <{A}{B}{C}> et <{A}{B}{D}>.

Le théorème suivant et le Théorème 3.1 garantissent que l'extension de l'arbre de préfixes PSP permet, pour toutes les longueurs k de séquences fréquentes trouvées, de générer l'ensemble de toutes les (k+1)−séquences candidates susceptibles d'être fréquentes.

Théorème 3.2 : [27]

Soit Fk l'ensemble des k−séquences fréquentes, la génération de séquences candidates par l'auto-jointure de Fk sur lui-même ou par l'extension de l'arbre de préfixes PSP, construit le même ensemble Ck+1 de séquences candidates de longueur (k+1).

2.1.2.3 Calcul des supports

Comme dans l'algorithme GSP, le calcul des supports des séquences candidates nécessite un parcours complet de la base de séquences de données en projetant chaque séquence lue, dans l'arbre de préfixes afin d'incrémenter les supports de seulement les séquences candidates qui son supportées par cette séquence. Toutefois, dans PSP, cette projection est réalisée, à la différence de GSP, de la manière suivante : PSP parcourt son arbre de préfixes guidé par la séquence de données lue jusqu'à ce que ce parcours échoue, signe de non inclusion des séquences candidates représentées par les feuilles qui sont descendantes du sommet de cet échec. Si le parcours réussit alors les séquences candidates représentées par les chemins de la racine vers les feuilles sur lesquelles l'algorithme aboutit, sont supportées par la séquence lue et les supports des feuilles sont par conséquent, incrémentés.

Application à un exemple

Reprenons notre base de séquences de données D' de la Figure 3.8. Le présent exemple explique l'algorithme PSP, appliqué à cette base de séquences de données. Nous considérons un seuil de support minimum de 2/3.

Itération (1) : Chaque item de D' est représenté par une feuille reliée à la racine par une branche de type Other transaction afin de construire C1, l'arbre de préfixes des 1−séquences candidates. Un parcours de D' avec la projection de toutes ses séquences dans cet arbre, permet de calculer leurs supports, pour ensuite, élaguer les feuilles dont les supports sont insuffisants afin de ne garder dans F1 que les 1−séquences fréquentes.

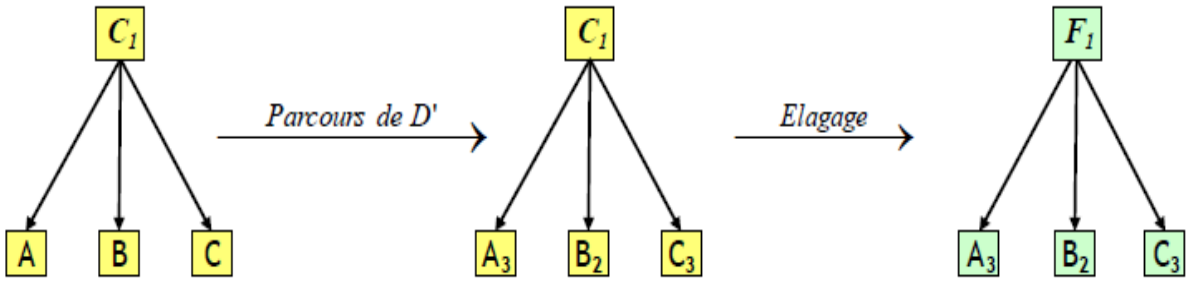


Figure 3.8–Séquence fréquente de longueur 1

Itération (2) : PSP procède à une génération de 2−séquences candidates similaire à celle dans GSP. L'auto−jointure F1 ⊗F1 permet de générer les 2−séquences candidates qui sont insérées dans l'arbre de préfixes C2 ci-dessous. Un second parcours de D' avec sa projection dans cet arbre, permet de calculer leurs supports, pour ensuite, élaguer les feuilles dont les supports sont insuffisants afin de ne garder dans F2 que les 2−séquences fréquentes.

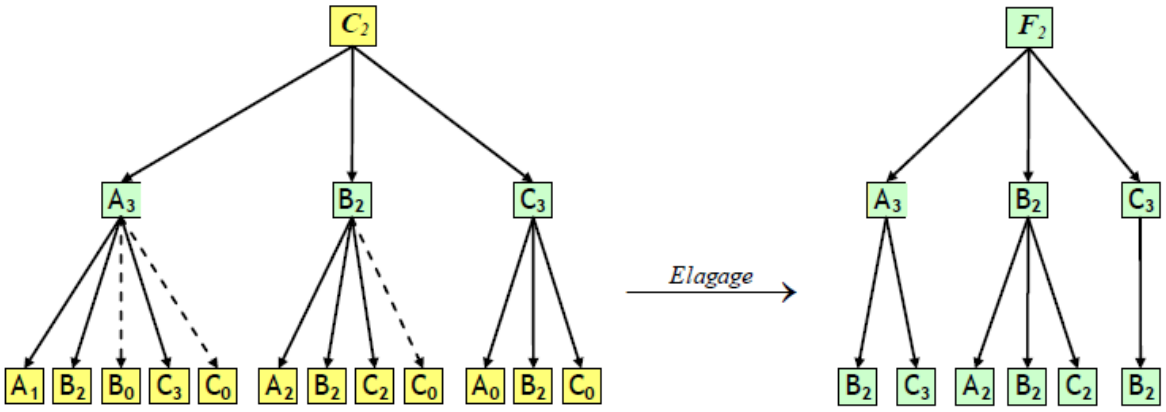


Figure 3.9 –Séquences fréquentes de 1 et 2

Itération (3) : L'extension PSP de l'arbre F2 permet de construire l'arbre C3 ci-dessous, représentant les 3−séquences candidates générées : <{A,B}{C}>, <{A}{C}{B}>, <{B}{C}{B}> et <{C}{B}{C}>. Remarquons que l'item B fils de A dans cet arbre a été étendu uniquement par l'item C qui est son fils de la racine tout en étant son frère de A. PSP s'assure ainsi de l'application de la propriété d'antimonotonicité dans sa génération de séquences candidates.

Une troisième projection des séquences de D' dans l'arbre C3, permet de calculer les supports de ses feuilles pour enfin, ne garder dans l'arbre F3 que les 3−séquences fréquentes.

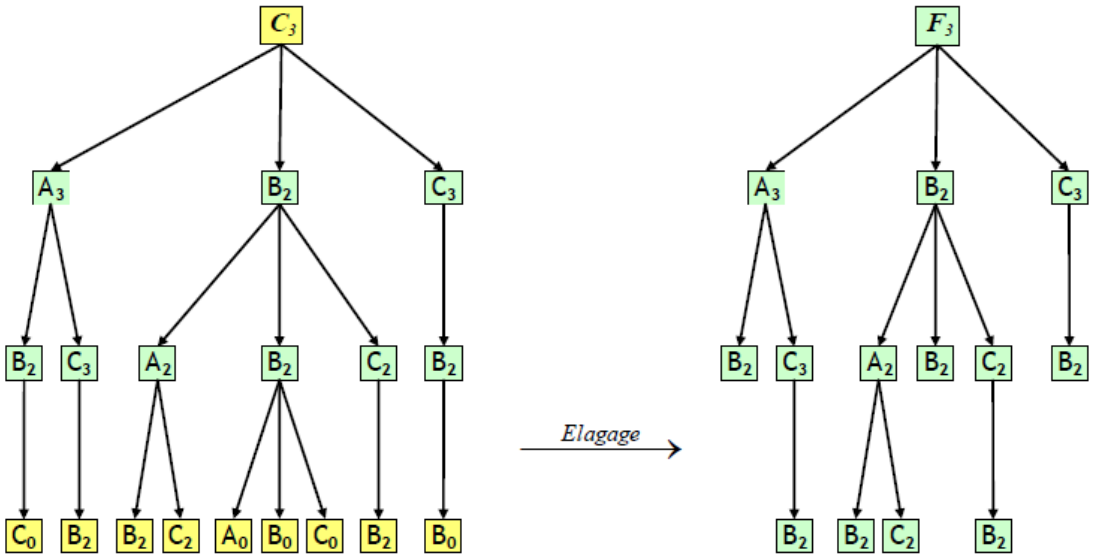


Figure 3.10–Séquences fréquentes de 1,2 et 3

Itération (4) : L'extension PSP de l'arbre F3 permet de générer la seule 4−séquence fréquente <{B}{A}{C}{B}> qui est de support 2 déterminé par une quatrième projection des séquences de D' dans C4. Cette séquence est donc gardée dans l'arbre F4 suivant.

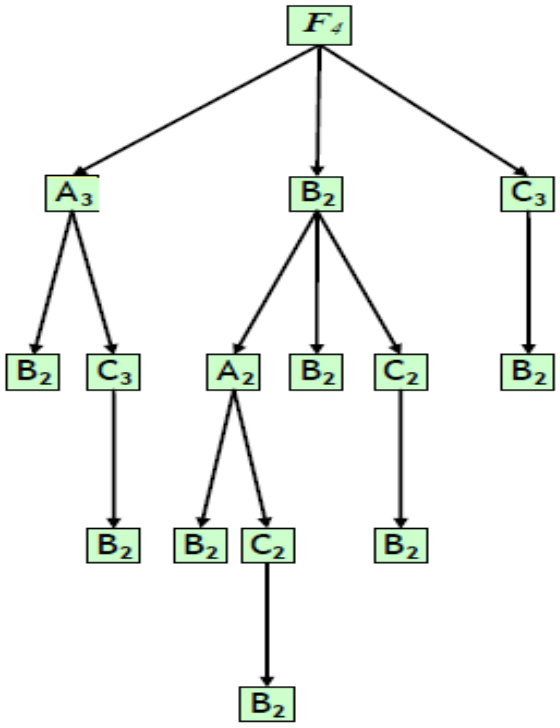


Figure 3.11–Séquences fréquentes de longueur 1,2,3 et 4

Itération (5) : L'extension de l'arbre F4 ne génère aucune 5−séquence fréquente. L'algorithme arrête la recherche et retourne en résultat, l'ensemble de toutes les séquences fréquentes trouvées, autrement dit, les 14 séquences au total représentées dans la dernière instance F4 de son arbre de préfixes, c'est à dire : <{A}>:3, <{B}>:2, <{C}>:3,<{A}{B}>:2,<{A}{C}>:3,<{B}{A}>:2,<{B}{B}>:2,<{B}{C}>:2,<{C}{B}>:2,<{A}{C}{B}>:2,<{B}{A}{B}>:2,<{B}{A}{C}>:2,<{B}{C}{B}>:2,et<{B}{A}{C}{B}2.

2.1.2.4 Discussion

L'une des caractéristiques de l'arbre de préfixes apporté par l'algorithme PSP et qui permet de le distinguer de l'arbre de hachage GSP, est, comme le montre aisément l'exemple précédent, le fait que les séquences candidates et les séquences fréquentes sont toutes représentées par cette même structure. PSP est bâti autour et en bénéficie efficacement, notamment, dans sa phase de génération de séquences candidates.

S'appuyant sur cette structure, l'algorithme réduit considérablement sa consommation en mémoire centrale, car l'arbre de préfixes sert à représenter directement les séquences candidates, à l'inverse des arbres de hachage qui sont pénalisés par le stockage inévitable d'une structure de clefs d'accès en sus. De plus, le nombre d'items utilisés dans cet arbre est généralement inférieur au celui que présentent les séquences candidates au total. Ceci est justifié par le fait que ce nombre est exactement de l'ordre du nombre d'items que présentent les séquences candidates moins celui des items appartenant à leurs préfixes communs.[26]

2.1.3 Algorithme SPAM [07]

Assez proche de SPADE, mais opérant une recherche en profondeur d'abord, l'algorithme SPAM **(Sequential Pattern Mining)** [38] gère le support d'une séquence candidate par une séquence de données à l'aide d'une représentation en vecteurs de bits de ces séquences.

Nous verrons dans cette section que cet algorithme suppose que toute la base de données doit initialement être chargée en mémoire centrale sous cette forme, également dite représentation bitmap verticale, et propose ainsi, une méthode de comptage de supports simple et efficace.

Quant à l'exploration de l'espace de recherche, SPAM se base essentiellement sur une structure en arbre lexicographique de séquences qu'il construit et parcourt à fur et à mesure de sa découverte de l'ensemble des séquences fréquentes.

2.1.3.1 principes de bases

SPAM considère l'existence d'un ordre lexicographique noté $''\geq''$, sur les items de la base de données : si l'item *j* apparaît après l'item *i* alors $j \geq i$. Cette relation d'ordre est étendue aux séquences en définissant Sb $\geq$ Sa si Sb est sur−séquence de Sa. En se basant sur ce principe, l'algorithme parcourt l'espace de recherche **en profondeur** à l'aide d'un *arbre lexicographique de séquences* respectant cette relation qu'il construit récursivement au fur et à mesure de l'avancement de son exploration de cet espace.

La racine de l'arbre représente la séquence vide notée ζ, tandis que chaque nœud dans l'arbre représente une séquence candidate, et de manière récursive, si *n* est un nœud, alors tous ses fils sont des nœuds *m* tel que $m \geq n$, et la branche qui relie *n* à *m* est telle que la séquence représentée par *m* est une *extension* de la séquence représentée par *n*. Cette extension peut être obtenue de deux manières : soit par ***S−extension*** (*Séquence extension*) en ajoutant un nouvel itemset constituée d'un seul item à la séquence du nœud *n*, soit par ***I- extension***(*Itemset extension*) en ajoutant un item au dernier itemset de la séquence de *n*, tel que cet item soi plus grand que n'importe lequel des items de cet itemset.

Exemple

La Figure 3.12 représente un arbre lexicographique de séquences complet pour deux items A et B avec 3 itemsets maximum par séquence. Elle montre que chaque niveau k de l'arbre correspond aux k−séquences candidates ordonnées dans l'ordre lexicographique.

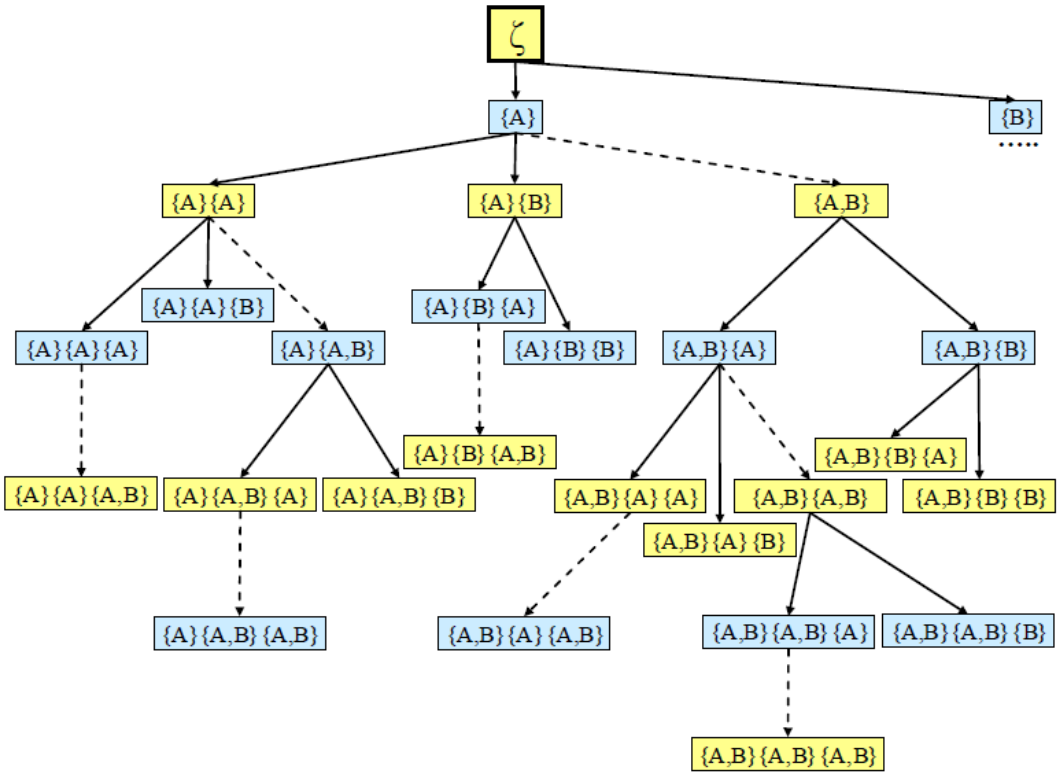


Figure 3.12 – Arbre lexicographique de séquences pour deux items A et B (Branche A seulement)

Les S−extensions sont représentées par des branches à trait plein (ce qui correspond à Other transaction dans PSP). Quant aux I−extensions, elles sont représentées par des branches en pointillées (Same transaction dans PSP). La séquence <{A,B}{A}{B}> est une S−extension de <{A,B}{A}> par l'item B, tandis que <{A,B}{A,B}> en est une I−extension.

Comme mentionné dans cette section, l'algorithme SPAM construit et parcourt cet arbre en profondeur, à fur et à mesure de l'avancement de sa recherche de motifs séquentiels. En résumé, ceci est réalisé de la manière suivante. Pour chaque nouveau nœud N de niveau k créé, sont générées des séquences candidates de longueur k+1 par S−extension et I−extension.

Les supports de ces séquences sont ensuite comptés. Nous verrons dans la section suivante que cette phase est grandement facilitée dans SPAM, grâce à une représentation en vecteurs de bits de ces séquences. Si le support d'une séquence générée est suffisant, autrement dit, elle est fréquente, alors est créé, un nouveau noeud N' de niveau (k+1) fils de N dans l'arbre, la représentant, et les étapes précédentes sont répétées récursivement sur N'. Dans le cas où aucune séquence générée n'est fréquente,

le noeud N est une feuille. SPAM quitte cette branche et remonte d'un niveau dans la récursivité vers le haut de l'arbre.

2.1.3.2 Représentation en vecteurs de bits verticaux des séquences candidates

Afin de minimiser les temps nécessaires aux opérations de génération de séquences candidates par S−extension et I−extension ainsi que les temps de calcul de leurs supports, SPAM apporte une représentation en vecteurs de bits verticaux de ces séquences. Dans sa première étape, l'algorithme crée en mémoire pour chaque item (1−séquence candidate) de la base de données, ce qui constitue l'unique parcours de cette dernière, un vecteur de bits vertical contenant un bit correspondant à chaque transaction de la base. Si un item i apparaît dans une transaction j d'un CID donné pour une Date donnée, alors le bit correspondant est mis à 1, autrement il a pour valeur 0. Le résultat est une représentation bitmap verticale de cette base, comme le montre la Figure 3.13 de l'exemple suivant.

Exemple

D'	**01**	**02**	**03**	**04**	**05**	**06**	**07**
C_1	{A,B,D}		{B,C,D}			{B,C,D}	
C_2		{B}		{A,B,C}			
C_3					{A,B}		{B,C,D}

L'unique parcours de D'

CID	***DATE***	**<{A}>**	**<{B}>**	**<{C}>**	**<{D}>**
C_1	01	**1**	**1**	0	**1**
	03	0	**1**	**1**	**1**
	06	0	**1**	**1**	**1**
C_2	02	0	**1**	0	0
	04	**1**	**1**	**1**	0
	–	0	0	0	0
C_3	05	**1**	**1**	0	0
	07	0	**1**	**1**	**1**
	–	0	0	0	0

Figure 3.13–Représentation SPAM en vecteurs de bits verticaux d'une base de séquences de données

La Figure 3.13 illustre la manière par laquelle l'algorithme SPAM représente la base de séquences de données D' par des vecteurs de bits verticaux correspondant aux items de cette base. Le parcours de D' lors de cette étape permet de connaître le nombre de valeurs de l'identifiant temporel (Dates) pour chaque CID. SPAM partitionne alors ces vecteurs en sections où chaque section représente les Dates d'un CID donné, ce qui permet à l'algorithme, dans sa phase de génération de séquences candidates (par S−extension et I−extension), de facilement déduire les vecteurs de bits correspondant à ces séquences ainsi que le comptage de leurs supports, grandement optimisé grâce à cette technique.

Les S−extensions : Soit un vecteur de bits V(s) représentant une k−séquence fréquente s et le vecteur de bits V(i) de l'item i. La S−extension de la séquence s par l'item i est réalisée comme suit. Un vecteur de bits V'(s) est tout d'abord généré à partir de V(s) : Pour chaque section CID dans V(s), soit j le premier bit dans cette section qui a pour valeur 1, les bits situés avant j dans cette section auront dans V'(s) pour valeur 0 et les bits situés après j, auront pour valeur 1. La S−extension est réalisée via l'opérateur ET logique entre ce vecteur de bits généré V'(s) et V(i), ce qui donne naissance à une nouvelle (k+1)−séquence candidate.

Exemple

Considérons notre représentation en vecteurs de bits et un seuil de support minimum de 2/3.

Le présent exemple illustre une S−extension de la 1−séquence fréquente s = < {A}> de support 3 par l'item B, ce qui permet de générer la 2−séquence candidate <{A}{B}>.

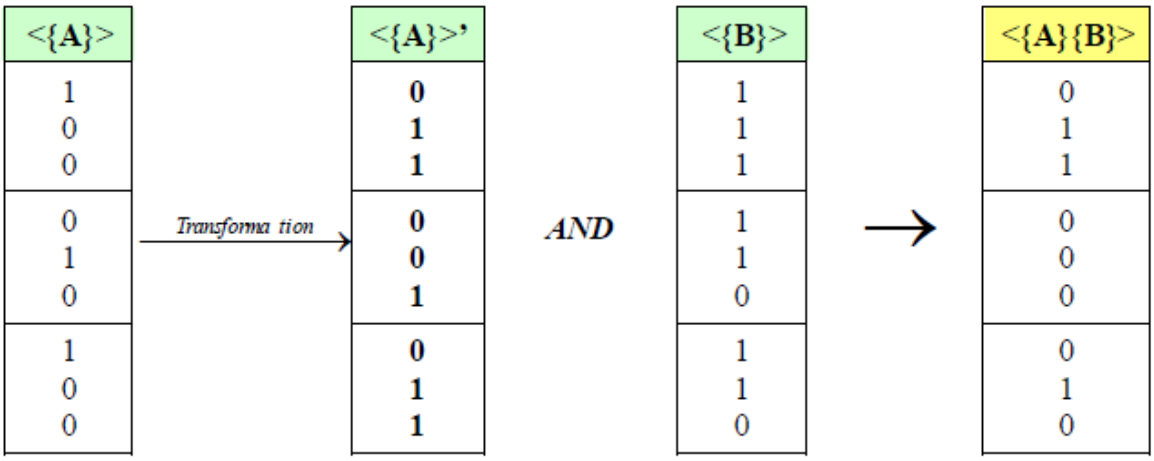


Figure 3.14 – S−extension de la séquence <{A}> par l'item B

Les I−extensions : Soit un vecteur de bits V(s) représentant une k−séquence fréquente s et le vecteur de bits V(i) de l'item i. La I−extension de s par i n'est autre que le vecteur de bits résultat de l'opérateur ET logique entre les deux vecteurs V(s) et V(i), ce qui donne naissance à une nouvelle (k+1)−séquence candidate, représentée par ce nouveau vecteur.

Exemple

La Figure 3.15 illustre une I−extension de la 2−séquence fréquente <{A}{B}> par l'item D, ce qui permet de générer la 3−séquence candidate <{A}{B,D}>.

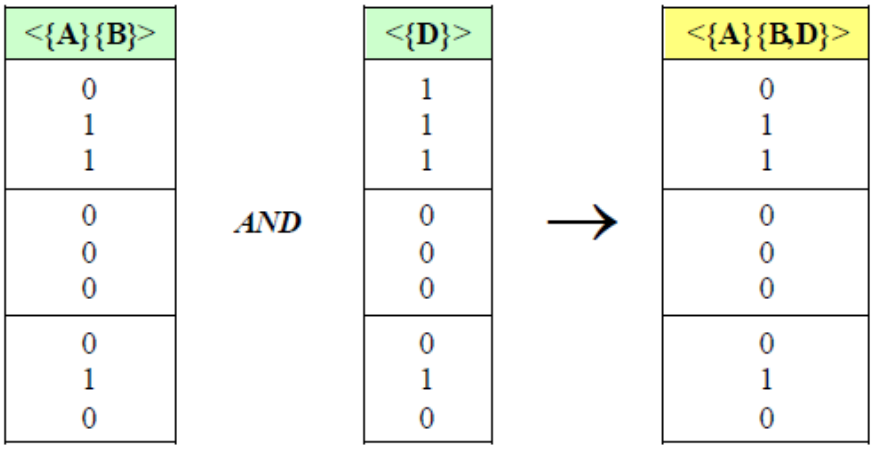


Figure 3.15–I−extension de la séquence < {A}{B}> par l'item D

Le processus de génération de séquences candidates par S−extension est appelé le S−step et celui de séquences candidates par I−extension, le I−Step. Il est ainsi possible d'associer à chaque noeud N de l'arbre deux ensembles : SN, l'ensemble des items candidats pour le S−step, et IN, l'ensemble des items candidats pour le I−Step, comme montré dans le pseudocode suivant, voir Procédure SPAM−Traverse−Noeud (N = <s1;s2;… ;sn>, SN, IN).

Algorithme 3.4 : *SPAM* (*D', MinSup*)

Extraction de motifs séquentiels dans une base de séquences de données *D'*, transformée d'une base de données temporelle, avec un seuil de support minimum *MinSup*

Entrée : Base de séquences de données *D'*; Liste d'attributs I; Support minimum *MinSup*;
Sortie : Ensemble des séquences fréquentes dans *D'*;
1. **Pour chaque** item $a \in$ I **faire** // Initialisation des vecteurs de bits des 1−séquences
2. **Début**
3. Créer *Vecteur*(<{*a*}>) = (0,0,…,0);
4. *Nbre−dates−différentes*<{*a*}> = 0; // Servira pour la partition des vecteurs de bits
5. **Fin;**
6. **Pour chaque** séquence de données $S \in D'$ **faire**
7. **Pour chaque** *Vecteur*(<{*a*}>) **faire** // Construction des vecteurs de bits des 1−séquences
8. **Pour chaque** itemset *T* dans *S* **faire**
9. **si** $a \in T$ **alors Début**
10. *Vecteur*(<{*a*}>)[*CID*(*S*), *Date*(*S*,*T*)] = 1;
11. Incrémenter *Nbre−dates−différentes*<{*a*}>;
12. **Fin;**
13. *Nbre−bits−par−section* = Max{*Nbre−dates−différentes*<{*a*}>, $a \in$I};
14. **Pour chaque** *Vecteur*(<{*a*}>) **faire**
15. **Début** // Partition des vecteurs de bits en section par *CID*
16.Etendre *Vecteur*(<{*a*}>) à (*Nbre−bits−par−section* ∗ *Nbre−CID*) bits;
17. Partitionner *Vecteur*(< {*a*}>) en *Nbre−CID* sections de *Nbre−bits−par−section*;
18. **Fin;**
19. **Pour chaque** *Vecteur*(<{*a*}>) **faire**
20. **Si** *Support−séquence*(*Vecteur*(<{*a*}>)) < *MinSup* **alors**
21. Supprimer *Vecteur*(<{*a*}>) // Purge des vecteurs des 1−séquences non fréquentes
22. **Sinon Début**
23. **Retourner** < {*a*}>;
// Initialisation de l'arbre lexicographique de séquences : branche <{*a*}>
24. Créér noeud $N \supset$ <{*a*}> fils *S−extension* de la racine;
// Ensemble *S*< {*a*}> des items candidats pour le *S−Step* (les *S−extensions*) de *N*
25.S_N = {$b \in$ I / $\exists$ *Vecteur*(<{*b*}>)};
// Ensemble *I*< {*a*}> des items candidats pour le *I−Step* (les *I−extensions*) *N*
26. I_N = {$b \in$ I / $\exists$ *Vecteur*(<{*b*}>) $\wedge$ b > a };
27. *SPAM−Traverse−noeud*(*N*, *SN*, *IN*);
28. **Fin;**

Procédure : *SPAM−Traverse−Noeud* (*N* = <*s*1;*s*2;… ;*sn*>, *SN*, *IN*)
Construction récursive de l'arbre lexicographique de séquences à partir du nœud *N* représentant la
séquence fréquente <*s*1;*s*2;… ;*sn*> en l'étendant par *SN* l'ensemble des items candidats pour les
S−extensions et *IN* l'ensemble des items candidats pour les *I−extensions*.

// Traitement des *S−extensions* de la séquence <*s*1;*s*2;… ;*sn*> par tous les items dans *SN*
1. **Pour chaque** item $a \in SN$ **faire**
2. **Si** *Nombre−sections−non−nulles*(Transformée (*Vecteur*(<s_1;s_2;… ;s_n>) ∧ *Vecteur*({*a*})) ≥ *MinSup*
3. **alors Début**
4. **Retourner** <*s*1;*s*2;… ;*sn*;{*a*}>;
5. Créér noeud *N'* ⊃ <s_1;s_2;… ;s_n;{*a*}> fils *S−extension* de *N* = <s_1;s_2;… ;s_n>;
6. $S_{N'}$ = {*b* ∈ I / ∃ *Vecteur*(<{*b*}>)}; // Uniquement les fréquents
7. $I_{N'}$ = {*b* ∈ I / ∃ *Vecteur*(<{*b*}>) ∧ b > a }; // Fréquents plus dans l'ordre ≥
8. *SPAM−Traverse−Noeud*(*N', SN', IN'*); // Appel récursif sur le nouveau nœud créé
9. **Fin;**

// Traitement des *I−extensions* de la séquence <*s*1;*s*2;… ;*sn*> par tous les items dans *IN*
10. **Pour chaque** item $a \in IN$ **faire**
11. **Si** *Nombre−sections−non−nulles*(*Vecteur*(<*s*1;*s*2;… ;*sn*>) ∧ *Vecteur*({*a*}) ≥ *MinSup* **alors**
12. **Début**
13. **Retourner** <*s*1;*s*2;… ;*sn* ⋃ {*a*}>;
14. Créér noeud *N'* ⊃ <*s*1;*s*2;… ;*sn* ⋃ {*a*}> fils *I−extension* de *N* = <*s*1;*s*2;… ;*sn*>;
15. *SN'* = {*b* ∈ I / ∃ *Vecteur*(<{*b*}>)}; // Uniquement les fréquents
16. *IN* ' = {*b* ∈ I / ∃ *Vecteur*(<{*b*}>) ∧ b > Max(*sn* ⋃ {*a*})}; // Fréquents + dans l'ordre ≥
17. *SPAM−Traverse−Noeud*(*N', SN', IN'*); // Appel récursif sur le nouveau noeud créé

Calcul des supports

Grâce à leur représentation en vecteur de bits partitionnés en sections correspondant chacune à un CID de la base, le calcul du support de n'importe quelle séquence candidate générée par un nœud fréquent dans l'arbre lexicographique de séquences (soit par S−extension ou par I−extension), est immédiat dans SPAM. En effet, ceci revient à simplement compter, le nombre de sections non nulles dans le vecteur de bits lui correspondant.

Exemple

Dans la Figure 3.14, le support de la 1−séquence fréquente <{A}>, selon son vecteur de bits, est de (1+1+1) = 3, ce qui est très bien vérifié dans la base de séquences de

données D'. De même pour <{B}> qui est de support (1+1+1) = 3. Cependant, la 2−séquence candidate <{A}{B}> générée par S−extension n'est supportée que (1+0+1) = 2 fois.

2.1.3.3 Discussion

L'algorithme SPAM peut être vu comme une autre contribution aux algorithmes basés sur des listes d'occurrences comme SPADE [28], mais avec un mode de représentation différent.

SPAM gère la présence ou l'absence d'un item dans une séquence de données par l'intermédiaire d'une représentation en vecteurs de bits verticaux des séquences.

Comme nous l'avons montré dans la section précédente, ceci permet à l'algorithme de bénéficier des opérations binaires très efficaces que permet cette représentation, notamment, dans sa phase de génération de séquences candidates. En effet, les S−extensions et les I−extensions sont ramenées à des ET logiques entre vecteurs de bits. Quant aux supports des séquences générées, ils sont efficacement donnés par les nombres de sections non nulles dans les vecteurs de bits correspondants. Toutefois, ces tâches peuvent être très coûteuses en temps de calcul pour le cas d'une base de données à très grand nombre d'items fréquents avec un nombre important de CID présents dans cette base.

Ce ne sont pas les seules différences, lorsqu'on rappelle que cet algorithme parcourt l'espace de recherche en profondeur à la recherche de séquences fréquentes, grâce à son arbre lexicographique de séquences qu'il construit à fur et à mesure de son avancement, ce qui lui donne tout l'avantage par rapport à SPADE qui le parcourt par niveau en calculant un nombre prohibitif de listes d'occurrences de séquences à chaque itération.

Le dernier point important concerne le nombre de parcours de la base de données qui est réduit dans l'algorithme SPAM à une seule lecture permettant sa représentation en mémoire centrale sous la forme bitmap avant toute extraction de motifs séquentiels. Toutefois, le prix à payer est une contrainte de limitation mémoire : Cette représentation qui malgré qu'elle permet de minimiser l'espace mémoire requis par rapport aux listes de SPADE, ne peut garantir que toute la base de données puisse tenir en mémoire centrale, sans compter l'espace requis par les vecteurs calculés dans les appels récursifs de cet algorithme.

2.2 Méthode verticale

2.2.1 Algorithme SPADE

Comme mentionné dans le dernier point de la discussion précédente, les temps de réponse des algorithmes de recherche par niveau étudiés jusqu'à présent sont très dépendants des opérations d'entrée/sortie que doit généralement effectuer ce type d'algorithmes, notamment, lorsqu'on sait que chaque phase de comptage de supports, aussi optimisée qu'elle soit, déclenche une lecture de toute la base de données (voir Section 2.1.1.3 et 2.1.2.3). Parmi les solutions proposées, est celle apportée par l'algorithme SPADE [28]. Tout comme exploité dans l'arbre de hachage et l'arbre des préfixes des deux algorithmes étudiés, les séquences présentent souvent des préfixes communs. SPADE exploite cette propriété autrement : En se basant sur ces préfixes communs, il regroupe les motifs séquentiels par classes d'équivalence et décompose ainsi le problème en sous problèmes qui seront traités en mémoire.

2.2.1.1 Principe de base

La solution apportée par SPADE consiste à n'effectuer, qu'une unique lecture de la base de séquences de données afin de représenter celle-ci en mémoire centrale sous la forme de listes d'occurrences de séquences, en vue que tous ses traitements ultérieurs s'effectueront sur ces listes. Ce type de représentation est formellement donné par la définition suivante.

Définition 01 (Occurrence d'une séquence)

L'occurrence d'une séquence S' dans une séquence de données S est définie par un triplet (CID, Début, Fin), où CID représente l'identifiant unique de la séquence de données S, alors que "Début" et "Fin" représentent l'information de localisation temporelle de S' dans S.

Exemple 1

La séquence de données S = <{A}{B}{B}{C}> peut être représentée sous la forme d'occurrences de ses sous séquences de longueur 1, comme suit : A(1,1,1), B(1,2,2), B(1,3,3) et C(1,4,4). Sa sous séquence de longueur 2, <{A}{C}>, serait représentée AC(1,1,4).

Exemple 2

Le présent exemple montre la manière par laquelle l'algorithme *SPADE* représente la base de Séquences de données *D'* de la *Figure 3.16* par des listes d'occurrences d'items.

D'	**01**	**02**	**03**	**04**
C_1	{A}	{B}	{B}	{C}
C_2	{C}	{A}	{B}	
C_3	{A}	{C}	{B}	
C_4	{C}	{A}	{B}	{A}
C_5	{B}	{A}	{D}	{A}

L'unique parcours de D'

***ID−LIST* = A**		
CID	*Début*	*Fin*
C_1	1	1
C_2	2	2
C_3	1	1
C_4	2	2
C_4	4	4
C_5	2	2
C_5	4	4

***ID−LIST* = B**		
CID	*Début*	*Fin*
C_1	2	2
C_1	3	3
C_2	3	3
C_3	3	3
C_4	3	3
C_5	1	1

***ID−LIST* = C**		
CID	*Début*	*Fin*
C_1	4	4
C_2	1	1
C_3	2	2
C_4	1	1

***ID−LIST* = D**		
CID	*Début*	*Fin*
C_5	3	3

Figure 3.16–Représentation SPADE en listes d'occurrences d'items d'une base de séquences de données

Ces listes représentent en fait, les occurrences des 1−séquences candidates <A>, <B>, <C> et<D> dans la base de séquences de données D'. Le comptage de leurs supports consiste alors à simplement regarder les cardinalités de ces listes, ce qui ne nécessite aucun parcours de D'.

Comme le montre la Figure 3.16, chaque liste est identifiée par un identifiant unique "ID−LIST" utilisé lors des opérations de jointure temporelle entre listes pour la génération de séquences candidates de longueurs supérieures, comme l'explique la section suivante.

2.2.1.2 Génération de séquences candidates

L'espace de recherche est subdivisé en classes d'équivalence. SPADE gère les séquences candidates et les séquences fréquentes à l'aide de ces classes. Deux

séquences de longueur k appartiennent à la même classe si elles présentent un préfixe commun de longueur (k−1).

La définition formelle d'une classe d'équivalence est donnée comme suit.

Définition 02 (Classe d'équivalence)

Soit Fk−1 l'ensemble des séquences fréquentes de longueur (k−1) et Pk−1(S) la séquence de longueur (k−1) qui préfixe une séquence fréquente S de longueur k. Une classe d'équivalence (k−1) est définie de la manière suivante :

$$[p \in F_{k-1}] = \{S \in F_k \mid P_{k-1}(S) = p\}$$

La génération des séquences candidates de longueur (k+1) est effectuée dans *SPADE* par des j*ointures temporelles* entre deux à deux, toutes les listes d'occurrences des k−séquences fréquentes appartenant à une **même classe d'équivalence** k, en d'autres termes, celles partageant le même préfixe de longueur (k−1).

L'algorithme *SPADE* introduit pour cette phase deux opérateurs :

– **La fusion** de deux k−séquences appartenant à la même classe d'équivalence, construit une (k+1)−séquence candidate. Formellement, si deux séquences $S1$ = <P;X> et $S2$ = <P;Y> partagent le même préfixe P, alors *Fusion* ($S1$, $S2$) = <P;X;Y>.

– **La jointure temporelle** des *ID−LIST* de deux séquences $S1$ et $S2$, notée *JoinT*($S1$,$S2$), est une nouvelle *ID−LIST*, tel que *JoinT*($S1$,$S2$) ←*ID−LIST* (*Fusion* ($S1$,$S2$)), avec : soit Z = *Fusion* ($S1$, $S2$). Pour chaque (*CID*1, $d1$, $f1$) dans *ID−LIST* ($S1$) et (*CID*2, $d2$, $f2$) dans *ID−LIST* ($S2$), satisfaisant *CID*1 = *CID*2 et $f1 < d2$, ajouter (*CID*1, $d1$, $f2$) à *ID−LIST* (Z).

Algorithme 3.1 : *SPADE* (*D', MinSup*)

Extraction de motifs séquentiels dans une base de séquences de données *D'*, transformée d'une base de données temporelle, avec un seuil de support minimum *MinSup.*

Entrée : Base de séquences de données *D'*; Liste d'attributs I; Support minimum *MinSup*;
Sortie : Ensemble des séquences fréquentes dans *D'*;

1. **Pour chaque** item $a \in$ I **faire** créer *ID−LIST*(<*a*>) = Ø;
2. **Pour chaque** séquence de données $S \in D'$ **faire**
3. **Pour chaque** *ID−LIST*(<*a*>) **faire** // Construction des listes d'occurrences d'items
4. **Pour chaque** itemset *T* dans *S* **faire**
5. **si** $a \in T$ **alors** Insérer (*CID*(*S*), *Date*(*S,T*), *Date*(*S,T*)) dans *ID−LIST*(<*a*>);
6. **Pour chaque** *ID−LIST*(<*a*>) **faire**
7. **Si** nombre *CID* distincts dans *ID−LIST*(<*a*>) < *MinSup* **alors**
8. Supprimer *ID−LIST*(<*a*>); // Purge des listes d'occurrences d'items non fréquents
9. $F1 = \emptyset$;
10. **Pour chaque** *ID−LIST*(<*a*>) **faire** $F_1 = F_1 \cup \{<a>\}$; // 1−séquences fréquentes
11. **Pour** (k=2; $Fk-1 \neq \emptyset$; k++) **faire**
12. **Début**
13. **Pour chaque** deux (k−1)−séquences S_1 et $S_2 \in F_{k-1}$ **faire**
14. **Si** *Préfixe* ($S1$, k−2) = *Préfixe*($S2$, k−2) **alors** // même classe d'équivalence
15. **Début**
16. Générer la séquence candidate $Z = S1[k-2];S1[k-1];S2[k-1]$; // Fusion
17. *ID−LIST*(*Z*) = *Jointure−temporelle*(*ID−LIST*(*S*1),*ID−LIST*(*S*2));
18. **Si** nombre *CID* distincts dans *ID−LIST*(<*a*>) < *MinSup* **alors**
19. Supprimer *ID−LIST*(<*a*>)
20. **Sinon** $F_k = F_k \cup \{Z\}$;// Séquence fréquente trouvée
21. **Fin;**
22. **Fin;**
23. **Retourner** $F = F_1 \cup F_2 \ldots \cup F_k$. // Ensemble des séquences fréquentes dans *D'*

Fonction : *Jointure−temporelle* (*ID−LIST*1, *ID−LIST*2)

Calcule la jointure temporelle de deux listes d'occurrences (produit une liste d'occurrences)

1. $ID-LIST_j = \emptyset$;
2. **Pour chaque** ($CID1$, $d1$, $f1$) $\in$ *ID−LIST*1 **faire** // ne nécessite aucun parcours de *D'*
3. **Pour chaque** ($CID2$, $d2$, $f2$) $\in$ *ID−LIST*2 **faire**
4. **Si** $CID_1 = CID_2 \wedge f_1 < d_2$ **alors** Insérer (CID_1, d_1, f_2) dans $ID-LIST_j$;
5. **Retourner** $ID-LIST_j$.

Application à un exemple

Afin d'illustrer l'effet des deux opérateurs *Fusion* et *Jointure temporelle* dans la génération de séquences candidates dans *SPADE*, nous proposons d'expliquer cet

algorithme appliqué à la base de séquences de données *D'* de la *Figure 3.16*, avec un seuil de support minimum de 2/5.

Aucune contrainte temporelle n'est considérée du fait que cette composante est traitée par son extension *C−SPADE* [29], qui sera présentée dans la ***Section II.2.1.3***.

Itération (1) : Comme montré dans la *Figure 3.16*, la base de séquences de données *D'* est parcourue son unique fois afin d'être représentée en mémoire centrale sous la forme de listes d'occurrences d'items, correspondant aux 1−séquences candidates <{A}>, <{B}>, <{C}> et

<{D}>. La vérification des cardinalités de ces listes (nombre de *CID* distincts ≥ 2) permet d'identifier les 1−séquences fréquentes : <{A}>, <{B}> et <{C}> et de ne garder par conséquent, que les listes correspondantes : *ID−LIST*= A, *ID−LIST*= B et *ID−LIST*= C.

Itération (2) : Afin de générer des 2−séquences candidates, *SPADE* calcule les jointures temporelles entre les listes d'occurrences des 1−séquences fréquentes identifiées à l'itération précédente. La vérification des cardinalités des listes obtenues (*Figure 3.12*), permet de trouver les 2−séquences fréquentes : <{A}{A}>, <{A}{B}>, <{A}{C}>, <{B}{A}>, <{C}{A}> et <{C}{B}>, pour enfin ne garder que les listes : *ID−LIST*= AA, *ID−LIST*= AB,ID−LIST= AC, ID−LIST= BA, ID−LIST= CA et ID−LIST= CB.

ID−LIST = AA

CID	Début	Fin
C_4	2	4
C_5	2	4

ID−LIST = AB

CID	Début	Fin
C_1	1	2
C_1	1	3
C_2	2	3
C_3	1	3
C_4	2	3

ID−LIST = AC

CID	Début	Fin
C_1	1	4
C_3	1	2

ID−LIST = BA

CID	Début	Fin
C_4	3	4
C_5	1	2

ID−LIST = BB

CID	Début	Fin
C_1	2	3

ID−LIST = BC

CID	Début	Fin
C_1	2	4
C_1	3	4

ID−LIST = CA

CID	Début	Fin
C_2	1	2
C_4	1	2
C_4	1	4

ID−LIST = CB

CID	Début	Fin
C_2	1	3
C_3	2	3
C_4	1	3

ID−LIST = CC

CID	Début	Fin
VIDE		

Figure 3.17– Jointures temporelles SPADE pour les 1−préfixes <{A}>, <{B}> et <{C}>

Itération (3) : La génération des 3−séquences candidates consiste à calculer les jointures temporelles entre les listes d'occurrences correspondant aux 2−séquences fréquentes de même classe d'équivalence, autrement dit, celles ayant le même préfixe de longueur 1, à savoir :

•***Classe* <{A}>**: La fusion des 2−séquences fréquentes <{A}{A}>, <{A}{B}> et <{A}{C}> (de même 1−préfixe <{A}>) calcule les jointures temporelles des 3−séquences candidates : <{A}{A}{A}>, <{A}{A}{B}>, <{A}{A}{C}>, <{A}{B}{A}>, <{A}{B}{B}>, <{A}{B}{C}>, <{A}{C}{A}>, <{A}{C}{B}> et <{A}{C}{C}>.

L'opération ne génère aucune liste de plus d'un élément.

•***Classe* <{B}>**: La fusion de la 2−séquence fréquente <{B}{A}> avec elle-même, calcule la jointure temporelle de la 3−séquence candidate <{B}{A}{A}>, ce qui donne une liste *ID−LIST*= BAA d'un seul élément, insuffisant pour qu'elle soit fréquente.

•***Classe* <{C}>**: La fusion des 2−séquences fréquentes <{C}{A}> et <{C}{B}> (de même 1−préfixe <{C}>) calcule les jointures temporelles des 3−séquences candidates : <{C}{A}{A}>, <{C}{A}{B}>, <{C}{B}{A}> et <{C}{B}{B}>. L'opération identifie la seule 3−séquence fréquente <{C}{A}{B}> montrée par le *Tableau 3.1*.

***ID−LIST* = CAB**		
CID	*Début*	*Fin*
C_2	1	3
C_4	1	3

Table 3.1 − Liste d'occurrences SPADE pour la 3−séquence fréquente <{C}{A}{B}>

Itération (4) : La fusion de la 3−séquence fréquente <{C}{A}{B}> avec elle-même, calcule la jointure temporelle de la 4−séquence candidate <{C}{A}{B}{B}>, ce qui donne une liste*ID−LIST*= CABB vide. L'algorithme arrête la recherche et retourne en résultat l'ensemble des 10 séquences fréquentes trouvées au total : <{A}>:5, <{B}>:5, <{C}>:4, <{A}{A}>:2, <{A}{B}>:4, <{A}{C}>:2, <{B}{A}>:2, <{C}{A}>:2, <{C}{B}>:3 et <{C}{A}{B}>:2.

2.2.1.3 Intégration de contraintes temporelles : C–SPADE [29]

Développé comme une extension très intéressante de SPADE, l'algorithme C–SPADE [40] permet de considérer dans sa recherche de motifs séquentiels, un ensemble de contraintes spatio−temporelles, en plus du seuil de support minimum. Ces dernières peuvent être spécifiées par l'utilisateur sous forme de paramètres, comme montré dans la Section 3.4 du chapitre précédent. Rappelons que les contraintes temporelles sur les motifs séquentiels son :

l'intervalle de temps [MinGap, MaxGap] et la fenêtre temporelle WinSize. Toutefois C−SPADE permet de spécifier une longueur temporelle maximale MaxLen des motifs extraits. Quant aux contraintes spatiales (non évoquées jusqu'à présent), C−SPADE permet d'exiger à l'avance la présence ou l'absence d'un ou plusieurs items dans ces motifs.

La prise en compte de ces contraintes dans C−SPADE est en fait assez intuitive. En effet, il Suffit d'appliquer un filtrage sur les listes d'occurrences générées à chaque itération, ou plus à l'avance, en les vérifiant pendant les opérations de jointure temporelle.

Formellement, soient (X, d1, f1) $\in$ID−LIST(S1) une occurrence de la k−séquence fréquente S1 dans une séquence de données X, et (X, d2, f2) $\in$ID−LIST(S2) une occurrence de la k−séquence fréquente S2 dans la même séquence X. Les occurrences dans X des (k+1)−séquences candidates générées par jointure temporelle doivent :

– Pour la contrainte MinGap : vérifier que $d2 - d1 \geq$ MinGap.

– Pour la contrainte MaxGap : vérifier que $d2 - f1 \leq$ MaxGap.

– Pour la contrainte MaxLen : vérifier que $f2 - d1 \leq$ MaxLen.

– Pour la contrainte d'absence d'un évènement : l'évènement non désiré est supprimé.

Il faut faire attention lors de la vérification de la contrainte MaxGap. En effet, une séquence non valide peut générer une séquence plus longue valide. Par exemple, si la contrainte est MaxGap = 2 et les séquences <{B}(2);{D}(4)> et <{B}(2);{F}(5)> sont fréquentes, la distance temporelle dans <{B}(2) ;{F}(5)> vaut 3, ce qui est supérieur à MaxGap Cependant, <{B}(2) ;{D}(4);{F}(5)> est une séquence candidate valide !

Afin de palier à ce problème, C–SPADE apporte une modification à la phase de génération de séquences candidates : Au lieu de joindre deux listes d'occurrences de classe k, il propose de joindre Fk avec F2. Ceci a toutefois l'inconvénient inévitable

d’affaiblir l'élagage à base du support [29], ce qui peut augmenter non seulement les temps d’extraction de l’algorithme mais également sa consommation en ressources mémoire.

2.2.1.4 Discussion

La solution apportée par SPADE consiste à n’effectuer qu’un unique parcours de la base de données afin de représenter celle-ci en mémoire centrale sous la forme de listes d'occurrences d’items, en vue que tous ses traitements s’effectueront exclusivement sur ces listes. Cette technique est sans doute très intéressante puisqu’elle permet de minimiser à son plus bas niveau le nombre d’opérations d’entrée/sortie que l’algorithme doit effectuer. Toutefois, elle présente l’inconvénient majeur que toute la base de données doit tenir en mémoire centrale sous cette forme, ce qui ne peut être assuré dans tous les cas.

Curieusement, les liste d’occurrences hormis le fait qu’elle permettent de simplifier le calcul des supports des séquences candidates en le restreignant au simple comptage des cardinalités des listes correspondantes, permettent de généraliser la notion de séquence (*Motif séquentiels généralisés)*, ce qui offre la possibilité d'extraire des motifs respectant un ensemble de contraintes. Ainsi, les algorithmes sur ces listes comme l’extension *C−SPADE*, permettent de gérer efficacement la majorité de ces contraintes.

Toutefois, cette représentation présente l’inconvénient des coûts élevés en temps UC des opérations de jointure temporelle entre listes, indispensables pour la génération de séquence candidates. Ce cas se présente, notamment, pour une base de données **très dense** ou lorsque les contraintes sur les motifs sont absentes ou moins restrictives.

En somme, le plus grand avantage de *SPADE* réside à notre connaissance, dans le fait qu’en se basant sur les préfixes communs que présentent les séquences, cet algorithme regroupe les motifs séquentiels par classes d’équivalence et décompose ainsi le problème en sous problèmes indépendants. Tout l'intérêt de cette approche est qu'elle permet la **parallélisation** des traitements, chaque classe d'équivalence pouvant être traitée séparément.

2.3 Méthodes par projection

2.3.1 Algorithme PREFIX−SPAN [20]

L'algorithme PREFIX−SPAN (Prefix projected Sequential Pattern mining) se base sur une étude attentive du nombre de séquences candidates qu'un algorithme de recherche par niveau peut avoir à générer afin de découvrir toutes les séquences fréquentes. L'objectif de son auteur [20] est alors de réduire le nombre de ces séquences générées. Pour cela, il propose d'exploiter, comme dans les trois algorithmes présentés : GSP, PSP et SPADE , les préfixes communs que présentent souvent les séquences de données.

Cependant, sa stratégie est si différente que l'algorithme ne génère aucune séquence candidate durant ses différentes phases de la recherche comme l'explique son principe de base.

2.3.1.1 Principe de base

PREFIX−SPAN se base sur les deux définitions suivantes afin de réaliser des projections successives de la base de séquences de données, servant à la partitionner en fonction des préfixes communs. Ceci permet à l'algorithme de diriger l'effort d'extraction vers les parties utiles de cette base, en d'autres termes, celles qui peuvent contenir des **motifs solutions**. Par exemple, si l'on ne souhaite extraire que les séquences qui commencent par le préfixe <AB>, cette méthode construira la projection de cette base, sur ce préfixe seulement, réduisant ainsi l'effort d'extraction en évitant de considérer la partie de la base qui n'est pas préfixée par la séquence <AB>. L'économie réalisée est donc intéressante.

Définition 03 (Projection d'une séquence)

Soient S et S' deux séquences de données avec S' $\subseteq$ S. On appelle projection de S sur S' la séquence de données P satisfaisant les propriétés suivantes :

(1) $P \lhd S$.

(2) $S' \prec P$.

(3) P est maximale, c'est-à-dire, $\forall$q une séquence,$(q \lhd S^*) \wedge (P \subset q) \Rightarrow S' \not\prec q$.

Définition 04 (Projection d'une base de séquences de données)

La projection d'une base de séquences de données D' sur une séquence de données S, est une nouvelle base de séquences de données $D'_{|S}$, contenant les projections de toutes les séquences contenues dans D' sur la séquence S.

Dans sa première étape, *PREFIX−SPAN* identifie tous les items (1–préfixes) fréquents de la base de séquences de données, ce qui nécessite un premier parcours de cette base, ensuite, il construit des bases intermédiaires, qui sont des projections de cette dernière sur chacun des 1−préfixes fréquents, ce qui constitue le deuxième et dernier parcours de la base de séquences de données. L'algorithme cherche à faire croître la longueur des motifs séquentiels découverts, en appliquant cette méthode de manière récursive : *PREFIX−SPAN* s'appelle alors récursivement sur chacune de ses projections. Ce principe est résumé dans le pseudo-code suivant, qui réécrit les grandes étapes de cette technique. [20]

Algorithme 3.2 : *Prefix−Span* (*D', MinSup, P*)

Extraction de motifs séquentiels dans une base de séquences de données *D'* avec un seuil de support minimum *MinSup*. Appel initial *PrefixSpan*(*D', MinSup*, ζ).

Entrée : Base de séquences de données *D'*; Liste d'attributs I; Support minimum *MinSup*;
Sortie : Ensemble des séquences fréquentes dans *D'*;
1. $k = P$;
2. $P_r = \{s:<P;q> / q \in I \wedge Sup(q) \geq MinSup\}$; // Génération des ($k$+1)−préfixes de *D'*
3. **Pour chaque** séquence de données $S \in D'$ **faire**
    4. **Pour chaque** (k+1)−préfixe $p \in Pr$ **faire**
        5. **Si** $p \subseteq S$ **alors** Incrémenter Sup(p); // Calcul des supports
6. **Pour chaque** (k+1)−préfixe $p \in Pr$ **faire**
    7. **Si** Sup(p) < *MinSup* **alors** Supprimer p de Pr; // Purge des préfixes non fréquents
8. **Pour chaque** (k+1)−préfixe $p \in Pr$ **faire**
    9. **Début**
        10. **Retourner** p;
            11. $D_p' = \emptyset$;// Initialisation de la projection de *D'* sur le (k+1)−préfixe p
    12. **Pour chaque** séquence de données $S \in D'$ **faire** insérer *Projection*(T,p) dans p D
        13. *Prefix−Span*($D_p' = \emptyset$, *MinSup*, p); // Appel récursif sur cette projection
14. **Fin;**

Fonction : *Projection* (*S, P*)
Calcule la projection d'une séquence de données *S* sur un préfixe fréquent *P*.

1. $k = 1$;
2. **Pour** $i = 1\ldots$ $\|P\|$ **faire**

3.Début

4. **Si** $i >$ $\|S\|$ **alors Retourner** ζ;
5. **Si** $P[i] = S[k]$ **alors** Incrémenter(k);
6. **Fin;**
7. **Retourner** $<P[1];P[2];\ldots;P[\ \|P\|\ ];S[k];S[k+1];\ldots;S[\ \|S\|\ ]>$.
3.

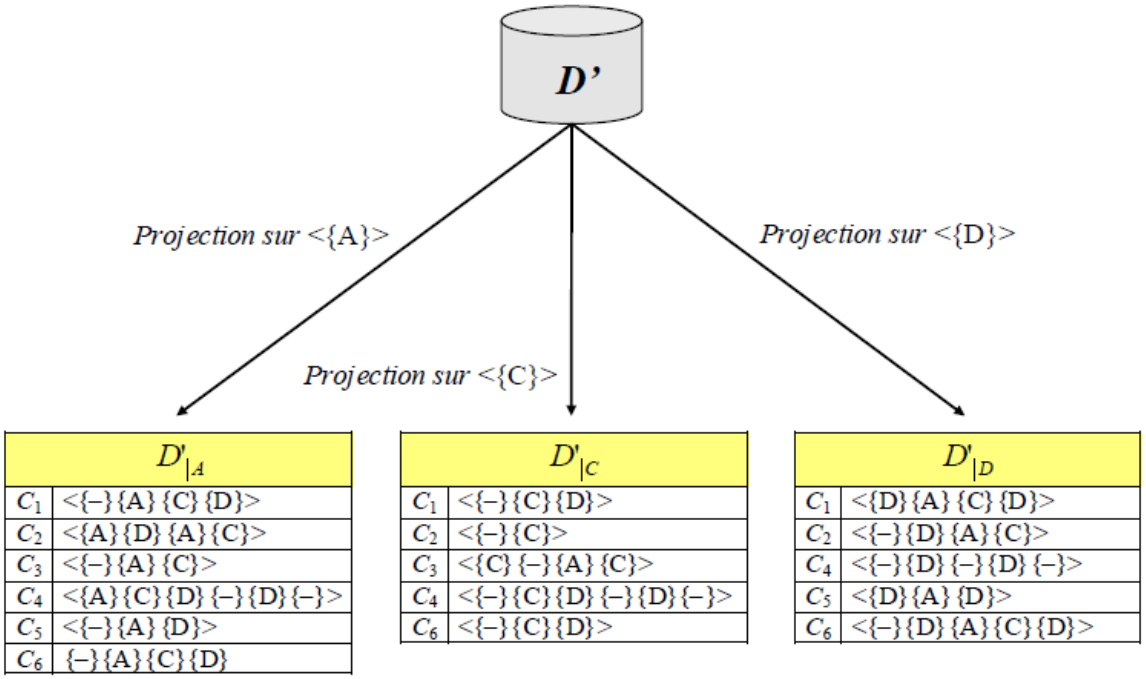
Application à un exemple

Cet exemple explique l'algorithme *PREFIX−SPAN* appliqué à la base de séquences de données *D'* de la table 3.2**,** Nous montrons notamment, la manière par laquelle l'algorithme opère des projections récursives de cette base afin d'extraire les motifs séquentiels sans génération de séquences candidates. Le support minimum considéré est de 3/6.

D'	**01**	**02**	**03**	**04**	**05**	**06**
C_1	{D}	{A}	{C}	{D}		
C_2	{A}	{D}	{A}	{C}		
C_3	{C}	{E}	{A}	{C}		
C_4	{A}	{C}	{D}	{B}	{D}	{F}
C_5	{D}	{A}	{D}			
C_6	{B}	{D}	{A}	{C}	{D}	

Table3.2 − Base de séquences de données exemple pour PREFIX−SPAN

Dans sa première étape, *PREFIX−SPAN* effectue un parcours de *D'* afin de calculer les supports de tous ses items, ce qui donne : <{A}>:6, <{B}>:2, <{C}>:5, <{D}>:5, <{E}>:1 et <{F}>:1. Les 1−préfixes fréquents sont donc : < {A}>, <{C}> et <{D}>. L'algorithme calcule alors, les projections de *D'* sur ces 1−préfixes seulement, c'est-à-dire : $D'_{|A}$,$D'_{|C}$ et $D'_{|D}$, montrées dans la *Figure 3.18*, Remarquons la disparition des items non fréquents qui sont représentés par le symbole "−".



$D'_{\|A}$	
C_1	<{–}{A}{C}{D}>
C_2	<{A}{D}{A}{C}>
C_3	<{–}{A}{C}>
C_4	<{A}{C}{D}{–}{D}{–}>
C_5	<{–}{A}{D}>
C_6	{–}{A}{C}{D}

$D'_{\|C}$	
C_1	<{–}{C}{D}>
C_2	<{–}{C}>
C_3	<{C}{–}{A}{C}>
C_4	<{–}{C}{D}{–}{D}{–}>
C_6	<{–}{C}{D}>

$D'_{\|D}$	
C_1	<{D}{A}{C}{D}>
C_2	<{–}{D}{A}{C}>
C_4	<{–}{D}{–}{D}{–}>
C_5	<{D}{A}{D}>
C_6	<{–}{D}{A}{C}{D}>

Figure 3.18 – Projections PREFIX–SPAN sur les 1–préfixes fréquents <{A}>, <{C}> et <{D}>

Exploration de $D'_{|A}$: L'algorithme effectue un parcours de cette projection et calcule les supports de tous les 2–préfixes commençant par <{A}>, c'est-à-dire : <{A}{A}>:0, <{A}{C}}>:5 et <{A}{D}>:2. Le seul 2–préfixe fréquent est <{A}{C}>. *PREFIX–SPAN* calcule la projection de $D'_{|A}$ ' sur ce 2–préfixe, notée $D'_{|A|AC}$. Dans cette projection, seul le 3–préfixe<{A}{C}{D}> est fréquent. La projection de D'|A|AC sur ce 3–préfix E, notée D'|A|AC|ACD, ne contient aucun 4–préfixe fréquent. L'algorithme quitte cette branche pour remonter d'un niveau dans la récursivité. PREFIX–SPAN aura trouvé pour cette branche les trois séquences fréquentes : < {A}>, <{A}{C}> et <{A}{C}{D}>.

$D'_{\|A}$	
C_1	<{–}{A}{C}{D}>
C_2	<{A}{D}{A}{C}>
C_3	<{–}{A}{C}>
C_4	<{A}{C}{D}{–}{D}{–}>
C_5	<{–}{A}{D}>
C_6	{–}{A}{C}{D}

→

$D'_{\|A\|AC}$	
C_1	<{–}{A}{C}{D}>
C_2	<{–}{A}{C}>
C_3	<{–}{A}{C}>
C_4	<{A}{C}{D}{–}{D}{–}>
C_6	{–}{A}{C}{D}

→

$D'_{\|A\|AC\|ACD}$	
C_1	<{–}{A}{C}{D}>
C_4	<{A}{C}{D}{–}{D}{–}>
C_6	{–}{A}{C}{D}

Figure 3.19 − Exploration PREFIX−SPAN de la branche projection sur le 1−préfixe fréquent < {A}>

Exploration de $D'_{|C}$**:** Cette projection contient le seul 2−préfixe fréquent <{C}{D}>. Sa projection sur ce 2−préfixe, notée *C CD D*' ne contient aucun 3−préfixe fréquent, ce qui termine l'exploration de cette branche. *PREFIX−SPAN* aura trouvé pour cette branche les deux séquences fréquentes : <{C}> et <{C}{D}>.

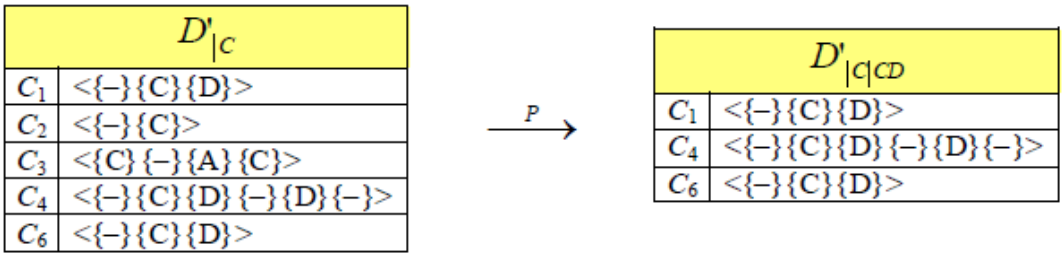


Figure 3.20 : Exploration PREFIX−SPAN de la branche projection sur le 1−préfixe fréquent <{C}>

Exploration de $D'_{|D}$**:** Cette projection contient le 2−préfixe fréquent <{D}{A}>, ce qui déclenche sa projection $D'_{|D|DA\ sur}$ ce 2−préfixe. Dans cette projection, seul le 3−préfixe<{D}{A}{C}> est fréquent. La projection $D'_{|D|DA|DAC\ sur}$ de $D'_{|D|DA\ sur}$ sur ce 3−préfixe ne contient aucun 4−préfixe fréquent, ce qui termine l'exploration de cette branche. *PREFIX−SPAN* aura $D'_{|D|DA}$ sur trouvé pour cette branche les séquences fréquentes : <{D}>, <{D}{A}> et <{D}{A}{C}>.

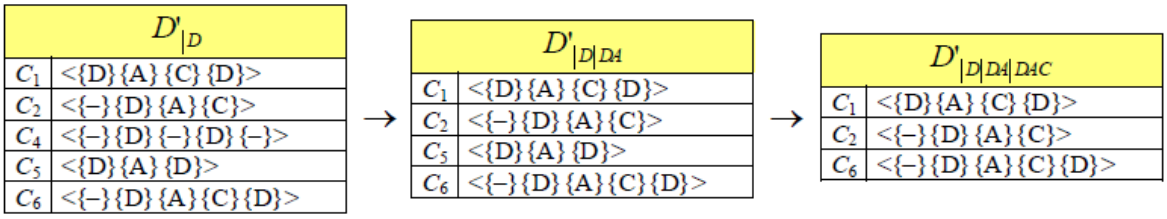


Figure 3.21 – Exploration PREFIX−SPAN de la branche projection sur le 1−préfixe fréquent <{D}>

Tous les niveaux de récursivité ont été parcourus, l'extraction est achevée. *PREFIX−SPAN* aura découvert au total les 8 séquences fréquentes : < {A}>:6, <{C}>:5, <{D}>:5,<{A}{C}>:5, <{C}{D}>:3, <{D}{A}>:4, <{A}{C}{D}>:3 et <{D}{A}{C}>:3.

2.3.1.2 Intégration de contraintes temporelles : C−Pattern−Growth

Les techniques de projection de la base de données, comme celle apportée dans la méthode *PREFIX−SPAN,* permettent de gérer facilement les contraintes de préfixe *monotones* et *antimonotones* (une classification de ces contraintes est donnée dans [30]). En effet, il suffit de restreindre dans la phase de calcul de leurs supports, les préfixes à seulement ceux qui satisfont la contrainte, pour réaliser l'élagage souhaité. Nous listons dans ce qui suit, le pseudo-code de l'algorithme *C−PATTERN−GROWTH* qui constitue une modification très intéressante de *PREFIX−SPAN,* permettant la considération de ces contraintes.

La conjonction *C* peut comporter à la fois des contraintes monotones et des contraintes antimonotones. Cependant, et comme nous l'avons mentionné dans ce mémoire, nous nous intéressons pour notre application à la maintenance industrielle, aux types de contraintes temporelles (intervalle de temps et fenêtre temporelle) qui sont antimonotones.

Algorithme 3.3 : *C−Pattern−Growth* (*D', MinSup, C, P*)
Extraction de motifs séquentiels avec une conjonction de contraintes antimonotones *C*
Appel initial *C−Pattern−Growth*(*D'*, *MinSup*, *C*, ζ).

Entrée : Base de séquences de données *D'*; Liste d'attributs I;
Support minimum *MinSup*; Conjonction de contraintes antimonotones *C*;
Sortie : Ensemble des séquences fréquentes dans *D'*;

1. $k = \|P\|$;
2. $Pr = \{s{:}\langle P;q\rangle / q \in I \wedge \mathrm{Sup}(q) \geq MinSup\}$;// Génération des (*k*+1)−préfixes de *D'*
3. **Pour chaque** séquence de données $S \in D'$ **faire**
4. **Pour chaque** (*k*+1)−préfixe $p \in Pr$ **faire** // Calcul des supports des (*k*+1)−préfixes modifié
5. **Si** $p \subseteq S \wedge C(p) =$ **VRAI alors** Incrémenter Sup(*p*);
6. **Pour chaque** (*k*+1)−préfixe $p \in Pr$ **faire**
7. **Si** Sup(*p*) < *MinSup* **alors** Supprimer *p* de *Pr*; // Purge des préfixes non fréquents
8. **Pour chaque** (*k*+1)−préfixe $p \in Pr$ **faire**
9. **Début**
10. **Retourner** *p*;
11. $D_p = \emptyset$;; // Initialisation de la projection de *D'* sur le (*k*+1)−préfixe *p*
12. **Pour chaque** séquence de données $S \in D'$ **faire** insérer *Projection*(*T*, *p*) dans *p D*' ;
13. *C−Pattern−Growth*(*p D*' , *MinSup*, *C*, *p*); // Appel récursif avec la conjonction *C*

14. **Fin;**

2.3.1.3 Discussion

L'algorithme *PREFIX−SPAN* issu d'une recherche en profondeur d'abord, propose d'effectuer des projections récursives de la base de séquences de données initiale en fonction des préfixes que présentent les séquences qu'elle contient (une démonstration du nombre de ces projections est donnée dans auparavant) . Ceci lui permet de réaliser des gains intéressants en temps de calcul en dirigeant l'effort d'extraction vers les parties utiles de cette base, autrement dit, celles qui peuvent contenir des motifs solutions. L'avantage principal de cette idée est que l'extraction peut se faire en **deux parcours seulement** de la base de séquences de données.

De plus, cette technique ne génère **aucune séquence candidate**. Toutefois, le prix à payer est une grande contrainte de limitation mémoire : Toutes les projections effectuées doivent tenir en mémoire centrale, ce qui ne peut être garanti dans tous les cas.

2.3.2 Free Span (Frequent pattern-projected sequential pattern mining)

Han et al (2000) ont introduit leur algorithme Free Span, dont l'idée est la suivante : Free Span projette la base de données séquentielle en base de données projetées. Chacune de ces derniers set nouveau récursivement projetée, de sort à que leur tailles diminues rapidement dans la majorité des cas (mais non obligatoirement), pour aboutir à la fin à des bases données plus petites, ce qui rend leur traitements plus facile. Cette méthode et considérablement plus rapide que celles basée sur l'heuristique Apriori (GSP).[25]

2.3.3 Les algorithmes LAPIN et PAID

Plusieurs nouveau algorithmes ont vu le jour à partir j.pei et j.han sur les méthodes par projection (pattern growth).Les auteures proposent l'algorithme LAPIN afin d'extraire des séquences fréquentes. Cet algorithme se base sur l'idée simple que seul la dernière position d'un item aa dans une séquence s permet de juger si une k-séquence fréquente peut être étendue avec l'item aa pour former une k+1-séquence.

Cette approche permet en fait de limiter encore plus l'espace de recherche parcouru lors de l'extraction de motif puisque l'algorithme ne s'intéresse qu'à construire une table

d'indexe des items pour chaque séquence dans la base de données. Une amélioration de l'algorithme LAPIN est proposée avec l'algorithme PAID qui s'appui sur une optimisation du précédent algorithme LAPIN en utilisant une représentation verticale de la base de données.

➢ Synthèse des différentes approches

Parmi les conclusions tirées de l'étude menée à travers ce chapitre, nous insistons sur le fait que l'espace de recherche exploré et le nombre de parcours de la base de données constituent pratiquement les deux facteurs clés de l'efficacité de tout algorithme d'extraction de motifs séquentiels. En se basant sur ces deux critères et d'autres comme la prise en compte de contraintes temporelles sur les motifs, nous proposons de présenter dans cette section une étude comparative des différentes approches étudiées tout au long de chapitre.

- **L'approche « Tester−et Générer »**

– La première proposition traitant les listes de transactions était l'algorithme **APRIORI−ALL** [16]. C'est une adaptation de l'algorithme APRIORI pour les séquences où la génération de séquences candidates et le calcul de leurs supports sont modifiés par rapport à la méthode de base. C'est pourquoi, APRIORI−ALL ne présente pas de grandes performances en temps de réponse ou en utilisation de l'espace mémoire.

– **L'algorithme GSP [**17] est similaire à APRIORI−ALL à la différence que les séquences candidates sont représentées plus efficacement dans son arbre de hachage. Toutefois, cette structure présente une faiblesse majeure qui consiste à mettre en commun des préfixes de séquences candidates sans faire de distinction entre deux items du même itemset et deux items avec changement d'itemset. De ce fait, l'algorithme se voit contraint de vérifier les séquences contenues dans les feuilles afin de savoir quels supports incrémenter.

– **L'algorithme PSP** [27] apporte une solution à cette faiblesse par son arbre de préfixes. S'appuyant sur cette structure, PSP réduit son utilisation de l'espace mémoire par le fait que les séquences sont directement représentées dans cet arbre. Toutefois, l'avantage qui permet de distinguer cette structure de l'arbre de hachage GSP réside dans le fait qu'elle permet une plus grande rapidité de détection des inclusions entre les séquences candidates contenues dans l'arbre et les séquences de la base de données.

Sachant qu'ils sont tous issus de l'approche **Tester−et−Générer**, ces trois algorithmes parcourent l'espace de recherche par niveau : pour chaque niveau k, un parcours de la base de données est nécessaire afin de calculer les supports. Si μ est la longueur de la plus longue séquence fréquente trouvée, alors on aura μlectures de cette base. Pour un μtrès grand, les performances peuvent être réduites très considérablement, notamment pour une base très dense ou lorsque les contraintes sur les motifs sont absentes ou moins restrictives.

– Comme solution à ce problème, **l'algorithme SPADE** [28] propose que toute la base de données soit initialement chargée en mémoire centrale sous la forme de listes d'occurrences de séquences. Cette solution présente toutefois un inconvénient qui réside dans les coûts UC des opérations de jointure temporelle entre listes à chaque génération de séquences candidates, sans compter la contrainte de limitation mémoire relative.

Il reste à discuter pour cette famille d'algorithmes, le nombre de séquences candidates générées à chaque niveau de la recherche. En effet, cette approche génère pour seulement n items fréquents, pas moins de $n^2 + \frac{n(n-1)}{2}$ séquences candidates de longueur 2. Ceci montre la taille énorme de l'espace de recherche à explorer pour l'extraction par niveau de motifs séquentiels. Cette complexité spatio−temporelle peut réduire considérablement les performances de ces algorithmes pour une base de données très dense.

- **L'approche « Diviser−pour−Régner »**

L'approche Diviser−pour−Régner propose une solution à ce problème en se basant sur une exploration en profondeur de l'espace de recherche. En effet, cette classe d'algorithmes permet de réduire ce facteur en divisant cet espace en sous contextes et en appliquant le processus de recherche récursivement sur chacun de ces sous contextes.

– **L'algorithme PREFIX−SPAN** [20] propose d'effectuer des projections récursives de la base de données en fonction des préfixes que présentent souvent les séquences, ce qui permet de réaliser des gains considérables en dirigeant l'effort d'extraction vers les parties utiles de cette base. Un autre avantage de cette idée réside dans le fait que l'extraction peut se faire en deux parcours seulement. De plus, la méthode ne génère aucune séquence candidate. Le prix à payer est une grande contrainte de limitation

mémoire : Toutes les projections doivent tenir en mémoire, ce qui ne peut être garanti dans tous les cas.

– **L'algorithme SPAM** [07] opère par une représentation de la base de données en mémoire centrale dans une structure en vecteurs de bits. Ensuite, il effectue l'extraction de *motifs séquentiels en exploitant un arbre lexicographique de séquences définissant son* espace de recherche. L'avantage principal est que l'algorithme n'effectue qu'un unique parcours de cette base, sans compter l'avantage des opérations binaires sur les vecteurs de bits, qui permettent une meilleure génération de séquences candidates ainsi que le comptage rapide de leurs supports. Toutefois, le prix à payer est le même que pour les deux algorithmes PREFIX−SPAN et SPADE : une grande contrainte de limitation mémoire.

Algo / ***Propriétés***	***GSP***	***PSP***	***SPADE***	***PREFIX-SPAN***	***SPAM***
Type de base de données	Base de séquences	Base de séquences	Base de transactions	Base de séquences	Base de transactions
Représentation des données	Horizontale	Horizontale	Verticale	Projections horizontales	Bitmap verticale
Localisation des données pendant l'analyse	Mémoire secondaire	Mémoire secondaire	Mémoire centrale	Mémoire centrale	Mémoire centrale
Structure de données	Arbre de hachage	Arbre de préfixes	Listes d'occurrences	Bases projetées	Vecteurs de bits
Représentation de l'espace de recherche	Aucune	Arbre de préfixes	Aucune	Aucune	Arbre lexicographique
Exploration de l'espace de recherche	*Breadth−first*	*Breadth−first*	*Breadth−first*	*Depth−first*	*Depth−first*
Technique	*Tester et Générer*	*Tester et Générer*	*Tester et Générer*	*Diviser pour Régner*	*Diviser pour Régner*
Génération de séquences candidates	Oui	Oui	Oui	non	Oui
Contraintes temporelles	Oui	non	non	non	non

Extensions pour la prise en charge de contraintes temporelles	−	Possible	*C−SPADE*	*C−PATTERN − GROWTH*	possible
Stratégie de parallélisme	Aucune	Aucune	Aucune	Aucune	Aucune
Perspectives de parallélisation	Faibles	Faibles	Faibles	Intéressantes	Intéressantes
Nombre de parcours de la base de données	Longueur maximale des motifs	Longueur maximale des motifs	1	2	1
Temps de réponse	+ + + + +	+ + + +	+ + +	+ +	+
Espace mémoire	+ +	+	+ + + + +	+ + + +	+ + +

Tableau 3.3 − Classification des algorithmes d'extraction de motifs séquentiels avec mention des propriétés les plus intéressantes

→Prise en charge de contraintes temporelles :

– *Dans GSP :* La prise en compte de contraintes temporelles est une composante principale de cet algorithme qui définit ses deux phases de recherche de séquences fréquentes sous contraintes : la phase avant et la phase arrière, sans compter que c'est le premier ayant posé la problématique des motifs séquentiels généralisés.

– *Dans SPADE :* L'ajout de contraintes temporelles est assez intuitif. En effet, la vérification de ces contraintes peut se faire en appliquant un filtrage sur les listes d'occurrences générées à chaque niveau ou en les vérifiant pendant les jointures temporelles.

– *Dans PREFIX−SPAN :*une extension de ce dernier peut facilement gérer les contraintes temporelles. En effet, cet algorithme effectue des projections récursives de la base de données sur les préfixes fréquents, il suffit donc de restreindre ces projections sur seulement les préfixes qui satisfont les contraintes pour réaliser l'élagage souhaité.

– ***Dans PSP et SPAM :*** A notre plus grande connaissance, il n'existe pas une extension de l'un de ces deux algorithmes dans le sens de l'intégration de contraintes temporelles.

Toutefois, nous pensons pour PSP, que puisque cette méthode est une recherche par niveau et que, par conséquent, un parcours de la base de données est nécessaire à chaque niveau, il est donc possible d'intégrer la vérification de ces contraintes lors de cette lecture afin de restreindre le comptage des supports à seulement les séquences les satisfaisant. Nous savons que c'est une tâche relativement complexe mais réalisable.

3. Conclusion

Dans ce chapitre, nous avons présenté un état de l'art des principales approches d'extraction de motifs séquentiels dans une base de données temporelle. Une étude comparative de ces approches ainsi que des différents algorithmes dans ce domaine ont été réalisés. Nous avons montré que l'application de ces algorithmes sur une même base de données avec les mêmes seuils de support donne les mêmes motifs. Il en découle que la comparaison doit concerner principalement les performances, en d'autres termes, les temps de réponse de ces algorithmes ainsi que l'espace mémoire requis pour la tâche d'extraction.

Les temps de réponse dépendent essentiellement du nombre de parcours de la base de données. En effet, il est clair que ces parcours en nombre très élevé peuvent générer une perte impressionnante de temps UC dans des opérations d'entrée/sortie. Toutefois, on ne peut minimiser l'importance des temps de traitements effectuées en mémoire lors des différentes phases de la recherche, allant de la génération de séquences candidates jusqu'au calcul de leurs supports. On peut conclure ici qu'il est primordial, pour une meilleure efficacité d'extraction, de viser l'objectif de réduire à son plus bas niveau possible l'espace de recherche parcouru. Quant à l'espace mémoire requis, il est clair que cet espace ne peut en aucun cas être estimé à l'avance pour une base de données non encore analysée, car il dépend essentiellement de sa densité, du nombre d'items ainsi que de la technique utilisée.

4

PSBI (Prefix tree for Sequential pattern mining using Bitmap representation)

1. Introduction

L'étude des différentes approches présentées et la comparaison des principaux algorithmes les implémentant ont montré que la prise en compte de contraintes temporelles en plus du seuil de support minimum dans l'extraction de motifs séquentiels n'est considérée originellement que dans l'algorithme *GSP* [17] . Pour le reste des algorithmes présentés, quelques extensions intéressantes ont été proposées, comme par exemple, pour *SPADE* [28], son extension *C−SPADE* basée sur le filtrage des listes d'occurrences générées à chaque niveau et *PREFIX−SPAN* [20] qui, son extension *C−PATTERN−GROWTH* restreint les projections de la base de données à seulement les préfixes qui satisfont ces contraintes pour réaliser l'élagage souhaité. Cependant, pour *PSP* [27] et pour *SPAM* [07], d'après la recherche que nous avons effectuée, il n'existe aucune extension dans ce sens de l'un de ces deux algorithmes. Toutefois, nous pensons à *PSP*, puisque cette méthode est une recherche par niveau et que, par conséquent, une lecture de la base de séquences de données est nécessaire à chaque niveau, il est donc possible d'intégrer le contrôle de ces contraintes lors de cette passe pour n'incrémenter les supports que des séquences les vérifiant.

C'est une tâche assez difficile vu les complexités relatives à son intégration dans le parcours de l'arbre de préfixes *PSP*, toutefois, elle est réalisable.

Partant du besoin pour notre exemple de la maintenance industrielle, de permettre aux spécialistes de la maintenance de spécifier les paramètres temporels qu'ils estiment optimaux afin de ne leur rapporter que les motifs les intéressant, en respectant par exemple des durées temporelles minimales et maximales bien définies entre pannes ou groupes de pannes découvertes .Nous proposons dans ce chapitre une solution de recherche de séquences de pannes fréquentes basée sur le principe *PSP*, enrichissant ce dernier par la prise en charge de contraintes temporelles dans l'extraction de ces motifs. Il est important de justifier notre choix d'extension de la méthode *PSP* par le fait qu'elle est particulièrement efficace pour les bases de données éparses par son principe de recherche en largeur (consommation mémoire réduite par rapport aux méthodes de recherche en profondeur). Toutefois, elle peut l'être pour une base de données dense, si

l'on arrive à *charger* cette dernière en mémoire centrale dans l'objectif de limiter le nombre de ses parcours coûteux.

En résumé, nos principales contributions dans ce chapitre sont :

–La présentation du domaine de la maintenance industrielle et ses différents types.

–La démonstration, vu l'absence de travaux d'extension de *PSP* dans le sens de la considération de contraintes temporelles, de la capacité de son arbre de préfixes d'intégrer un tel concept et son application (traitement d'un exemple) à un domaine très exigeant comme celui de la maintenance industrielle.

– Ne se contenter du principe de recherche par niveau de cet algorithme, que dans le sens où nous visons une stratégie qui n'effectue qu'un unique parcours de la base de données temporelle initiale, dans l'objectif de minimiser le temps de réponse.

– Minimiser l'espace mémoire et les temps UC relatifs aux opérations de génération des séquences candidates et au comptage de leurs supports par une représentation en vecteur de bits de ces séquences inspirée de la représentation bitmap de l'algorithme *SPAM*.

– Apporter des modifications aux optimisations de *PSP*.

Ce chapitre présent nos contributions pour une solution hybride proposée sous le nom ***PSBI* (*Prefix tree for Sequential pattern mining using Bitmap representation*)** tout en l'appliquant à la fonction de maintenance industrielle. Il est organisé comme suit : La section 2 présente le domaine de la maintenance industrielle, La section 3décrit la problématique de l'algorithme *PSP*. La section 4 introduit nos propositions pour l'amélioration de cet algorithme. La section 5 décrit notre algorithme de recherche de motifs séquentiels sous contraintes (*PSBI*) et la nouvelle structure de donnée proposée (*BVPT*). La section 6 présente un exemple qui illustre l'algorithme proposé sur une BDD de la maintenance, et la section 7 récapitule nos résultats. Enfin, une conclusion sur des points importants clôture ce chapitre.

2/La maintenance industrielle, l'assureur de production en entreprise

Métier historique mais officialisé par l'industrialisation depuis une quarantaine d'année, la maintenance industrielle se définit par le maintien ou la réparation d'équipements et moyens afin d'assurer une activité de production. Les missions d'un technicien résident autour d'actions de dépannage, réparation, vérification, contrôle, déclassement, réforme et gestion. Influée par le développement des technologies et les

nouveaux systèmes organisationnels, la maintenance industrielle dépasse sa fonction première pour s'afficher en tant qu'acteur majeur de l'amélioration de la qualité de la gestion de production des entreprises.

2.1/Les types de la maintenance industrielle

La maintenance corrective :travail de maintenance effectué après la détection d'une panne entraînant des arrêts de production. Il va s'agir d'un dépannage (maintenance palliative) ou d'une réparation (maintenance curative).

La maintenance préventive :travail de maintenance effectué dans l'intention de réduire la probabilité d'une panne. Il peut s'agir d'inspection, de contrôle ou de visites sur équipements.[31]

La maintenance prédictive : Ce mode de maintenance est basé sur l'application de techniques de mesure sur les équipements en service. Le but étant de diagnostiquer l'état d'un organe ou d'un équipement afin de juger l'opportunité de lancer l'opération de maintenance préventive ou de la reporter sur des bases rationnelles. Les techniques de mesure se basent généralement sur :

– L'analyse de l'état des équipements (fuite, fissure, corrosion, ancrage, …etc.)

– L'analyse des paramètres de procédé (température, pression, débit, …etc.)

– L'analyse des vibrations et bruits générés par les organes des machines tournantes.

– La vérification des huiles de lubrification et d'étanchéité.[32]

Comment peut-on améliorer le processus de la maintenance en se basant sur les données d'historique afin de mieux prévoir les pannes et minimiser les couts ?

La réponse est sans doute dans les données elles-mêmes. En effet, une analyse des données enregistrées au niveau de notre base de données, peut fournir de très bons résultats sous forme de connaissances nouvelles pour la maintenance.

En somme, nous nous intéressons dans notre exemple à la recherche de relations temporelles entre pannes et défauts des organes et équipements des installations pétrolières dans le but d'identifier des séquences particulières de ces pannes intéressant les spécialistes en maintenance. Notre objectif final est de permettre à ces derniers d'agir à l'avance sur des types de pannes jugées très coûteuses ou à risque sur la sécurité. Il s'agit d'un mode de maintenance plus que préventif, c'est la maintenance anticipée.

2.2. Notre Objectif : la Maintenance Anticipée (MA)

Tout simplement, la maintenance anticipée complète la maintenance traditionnelle dans le sens où elle intègre les techniques de Data mining qui permettent d'agir sur les pannes et défauts des équipements plus efficacement que le mode de maintenance préventive classique. Toute la force de la démarche que nous imaginons réside dans le fait qu'elle permet à l'opérateur en maintenance, d'intervenir au moins, avec une certitude connue à l'avance.

Notre vision d'une telle dotation vient du fait qu'il est très intéressant de concevoir et d'impliquer un outil d'extraction de connaissances qui permet de trouver des relations d'ordre temporel entre les différentes pannes et défauts des équipements industriels. La technique que nous adoptons à cet effet est celle des motifs séquentiels.

La démarche toute entière peut être vue alors comme un mariage entre la base de donnée, la maintenance industrielle et les motifs séquentiels. En effet, en se basant sur l'historique des pannes d'organes ou d'équipements, nous envisageons extraire des séquences de pannes qui se succèdent avec une certaine certitude, ce qui permet au système de réduire énormément les coûts de la maintenance et les temps d'arrêts non souhaités et dans certains cas, peut aider à sauver des vies humaines, surtout lorsqu'il s'agit de défaillances à risque sur la sécurité (très probables dans cette activité).

Considérons un exemple pratique :

Chaque fois que le joint X (dont le prix est de 900 DA) tombe en panne, le compresseur C (dont le prix dépasse les 900.000 DA) tombe en panne après 15 jours ! Dans ce cas une panne de quelques dinars engendre des pannes de plusieurs milliers de dinars ! Les opérateurs devront donc donner l'importance à la pièce X en la changeant régulièrement même avant l'expiration de son délai de maintenance préventive.

3. Problématique de l'algorithme PSP

PSP est un algorithme basé sur le principe *Tester−et−Générer*, mais qui utilise une organisation efficace des séquences candidates. Toute sa force réside dans son arbre de préfixes qui permet de minimiser les temps d'identification des inclusions des séquences candidates dans les séquences de données, par la projection directe de la base de séquences de données dans cet arbre, en plus de sa représentation à la fois des séquences candidates et des séquences fréquentes dans cette même structure. L'algorithme est bâti autour et en bénéficie efficacement, notamment, dans sa phase de

génération de séquences candidates. Toutefois, *PSP* présente quelques faiblesses que nous résumons comme suit :

Le nombre important de parcours coûteux de la base de séquences de données.

Le nombre important de séquences candidates générées à chaque itération.

La non considération de contraintes temporelles dans les motifs extraits.

(a)Le nombre important de parcours coûteux de la base de séquences de données

Comme pour toute approche basée sur le principe *Tester−et−Générer*, le problème réside dans le fait que *PSP*, par sa recherche par niveau, est contraint à chaque itération, d'effectuer un parcours complet de toute la base de séquences de données afin de calculer les supports des séquences candidates. Si μ est la longueur de la plus longue séquence fréquente trouvée, alors l'algorithme aura effectué μ parcours de cette base (Voir *Figure 4.1*). Pour un μ très grand, les performances en temps de réponse peuvent se dégrader très considérablement du fait de l'énorme utilisation dans ce cas de temps UC dans des opérations d'entrée/sortie.

Ce cas se présente, notamment, pour une base de données très dense ou lorsque le seuil de support minimum spécifié par l'utilisateur est trop faible.

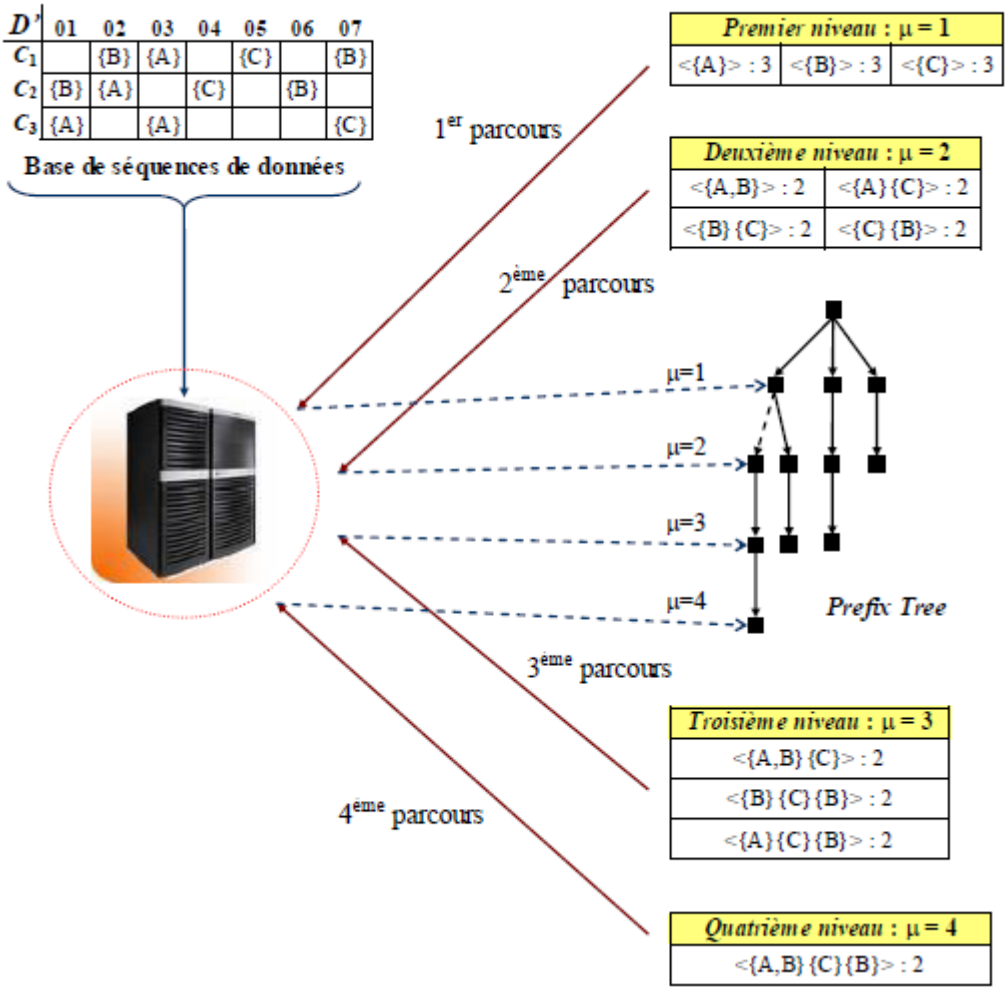


Figure 4.1 – Nombre important de parcours coûteux de la base de séquences de données dans l'algorithme PSP

(b)Le nombre important de séquences candidates générées à chaque itération

Comme dans tout algorithme de recherche par niveau (*GSP* [17], *SPADE* [28], …etc.), le problème réside dans le nombre important de séquences candidates à considérer à chaque itération vu la taille énorme de l'espace de recherche dans le cas des motifs séquentiels. Comme montré au chapitre précédent, le nombre de différentes séquences candidates à considérer est de

$$\psi_s = \sum_{k=1}^{m} (2^n - 1)^k = \frac{(2^n - 1)^{m+1} - 2^n - 1}{2^n - 2} = \Theta(2^{nm})$$

Avec n le nombre d'items de la base de séquences de données et m le nombre d'itemsets maximum dans une séquence de cette base. Cette complexité

spatio−temporelle nécessite une solution d'extraction qui, étant donné le grand écart entre le nombre de séquences candidates et celui des séquences fréquentes, doit déterminer quelles séquences candidates considérer, puis trouver efficacement lesquelles parmi celles-ci sont fréquentes.

c) La non considération de contraintes temporelles dans les motifs extraits

Ce qui se rencontre dans le cas d'une très grande base de données ou lorsque le seuil de support minimum est trop faible, ne peut permettre de cibler les connaissances extraites afin de restreindre la recherche à seulement ce qui intéresse l'utilisateur, et de limiter par conséquent, le nombre de motifs inutilement rapportés à ce dernier. Cette composante est justement d'extrême importance pour notre exemple, du fait que nous nous sommes fixés dès le début l'objectif d'offrir aux spécialistes de la maintenance le moyen efficace leur permettant de formuler une description à priori des motifs espérés (*distances temporelles minimales et maximales exigées entre pannes fréquentes et distance temporelle maximale entre pannes fréquentes dans un groupe de pannes*) par le biais des paramètres de cet outil, indispensable à notre connaissance, pour toute bonne application d'extraction de motifs séquentiels.

4. Améliorations de l'algorithme PSP

Partant des insuffisances discutées dans la problématique de l'algorithme *PSP*, nous proposons dans cette section de faire un pas en avant dans la présentation pour chaque préoccupation d'une solution, afin d'aboutir dans la prochaine section à l'algorithme proposé pour l'extraction de motifs séquentiels sous contraintes. Commençons par le problème (a) : le nombre de parcours coûteux de la base de données.

4.1. Réduction du nombre de parcours coûteux de la base de données

Les parcours en nombre important sur n'importe quelle base de données en entrée d'un algorithme d'extraction de motifs séquentiels sont tellement coûteux en temps UC que de nombreux travaux se sont particulièrement orienté vers la solution qui consiste à effectuer un pré-chargement de cette dernière en mémoire centrale avant toute recherche de ces motifs. Citons dans l'ordre, parmi les méthodes présentées dans ce mémoire :

1. L'algorithme *SPADE* [39] qui consiste à représenter la base de données en mémoire sous forme de listes d'occurrences d'items avant toute recherche.

2. L'algorithme *PREFIX−SPAN* [29] qui opère des projections en mémoire centrale de cette base selon les préfixes que présentent les séquences qu'elle contient.

3. L'algorithme *SPAM* [38] qui se base sur une représentation bitmap verticale en mémoire centrale de cette base afin de réaliser sa recherche en profondeur.

Les différentes discussions menées dans le chapitre précédent ont permis de conclure que la représentation *SPAM* est la moins consommatrice en espace mémoire du fait que l'occurrence d'une séquence candidate dans une séquence de données *CID* pour une valeur de l'identifiant temporel *DATE* est définie par un seul bit ayant la valeur 1 ou 0, représentant sa présence ou son absence dans cette séquence, ce qui est moins important que pour les bases projetées en mémoire de *PREFIX−SPAN* (notamment celles sur les préfixes les plus courts) ou les listes d'occurrences de *SPADE* (dans ses premières itérations, en plus des listes générées par les jointures temporelles, très coûteuses en espace mémoire).Ceci permet d'orienter notre choix vers la représentation bitmap verticale de la base de données afin de permettre de réaliser notre objectif de ne parcourir cette dernière qu'une unique fois pendant toute l'extraction. La proposition suivante montre la manière.

Optimisation 1 : Représentation en vecteurs de bits des items de la base de données ainsi que des occurrences des séquences candidates générées dans l'arbre de préfixes.

L'arbre de préfixes *PSP* permet de factoriser les séquences candidates selon leurs préfixes tout en considérant les changements d'itemsets dans cette factorisation (branches *Same transaction* et *Other transaction*). Notre idée principale consiste alors à exploiter cet arbre comme notre unique espace de recherche, à l'instar de l'arbre lexicographique de la méthode *SPAM*(dont le principe d'extraction est différent), où nous proposons de :

– Représenter les items fréquents de la base de données dans le premier niveau de l'arbre *PSP* par l'extension de chaque noeud de ce niveau par un vecteur de bits vertical représentant ses occurrences dans cette base.

– Pour les niveaux supérieurs, nous proposons une extension similaire permettant de représenter les occurrences dans la base de chaque *k*−séquence candidate générée par un vecteur de bits calculé pour ces occurrences.

– Le résultat est une représentation minimale par rapport aux méthodes *SPADE,PREFIX−SPAN* et *SPAM* de la base de données dans l'arbre de préfixes, permettant de transférer l'information complète sur les occurrences de tous ses items en un seul parcours de cette dernière, dans la nouvelle structure de données hybride ainsi définie que nous appelons *BVPT* (*Bit Vector Prefixed Tree*).

– Grâce à ce couplage entre la représentation de séquences candidates inspirée de *PSP* et la représentation bitmap verticale de la base de séquences de données, *BVPT* permet de représenter à la fois l'espace de recherche, l'ensemble de toutes les séquences candidates générées ainsi que l'ensemble de toutes les séquences fréquentes découvertes avec leurs supports. Quant aux séquences fréquentes maximales, elles sont facilement dérivées de la *BVPT* en identifiant les chemins maximaux dans cette structure.

La *Figure 4.2* montre une représentation en mémoire centrale de la base de séquences de données *D'*dans la structure *BVPT*. Ceci est réalisé en un seul parcours de *D',* qui constitue l'unique parcours de cette dernière pendant toute la recherche de séquences fréquentes du fait que l'information sur les occurrences et donc sur les supports des différentes séquences est très bien préservée dans cette nouvelle structure.

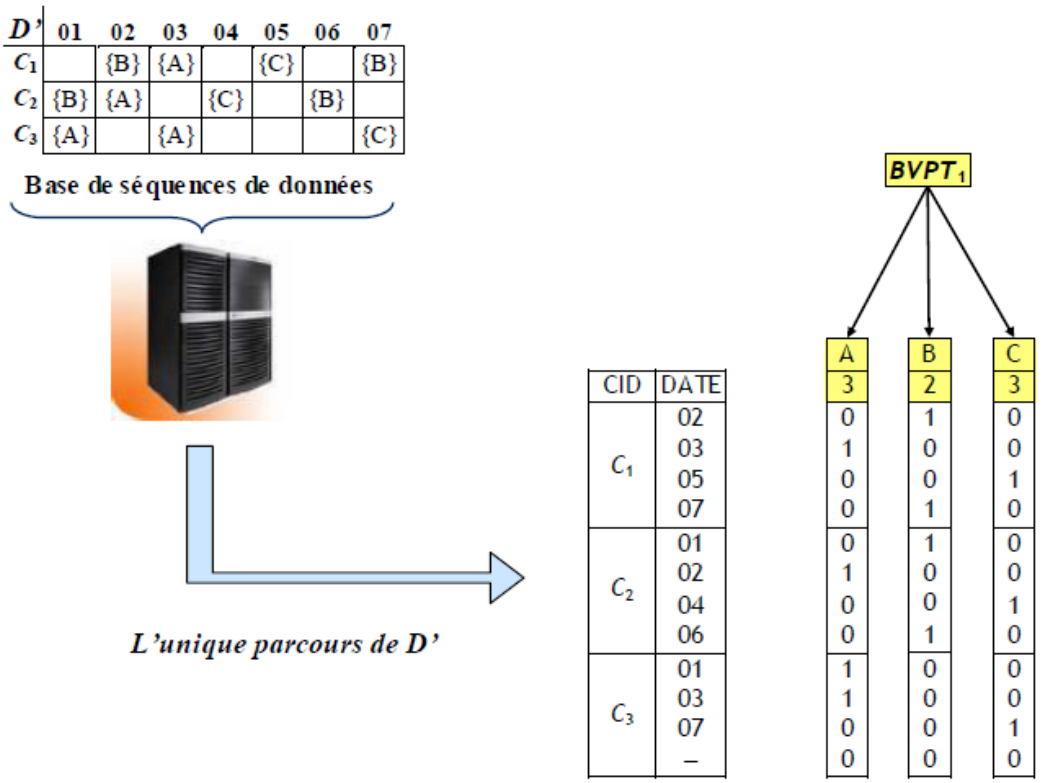


Figure 4.2 – Projection de la base de séquences de données D'dans la nouvelle structure BVPT (niveau 1)

– Remarquons que les vecteurs de bits verticaux sont partitionnés chacun en 3 sections où chaque section correspond à une séquence de données *CID* dans *D'*. Ceci permet d'appliquer les extensions *SPAM* permettant de facilement déduire les vecteurs correspondant aux séquences candidates générées par les opérations binaires très

efficaces que permet cette représentation et de minimiser par conséquent les temps UC nécessaires.

– Pour le calcul des supports des séquences candidates, *BVPT* permet à la différence de l'arbre *PSP* (nécessitant un autre parcours de *D'*), de simplement compter le nombre de sections non nulles dans les vecteurs correspondants : support immédiat

– *BVPT* apporte un gain intéressant en espace mémoire par rapport à l'arbre lexicographique de la méthode *SPAM*. Ce gain est donné par la réduction significative de la consommation en mémoire centrale des séquences candidates, justifiée par le fait que le nombre d'items utilisés dans la *BVPT* est de l'ordre du nombre d'items que présentent ces séquences moins celui des items appartenant à leurs préfixes communs.

Optimisation 2 : Elimination de la phase de transformation de la base de données temporelle initiale (D) en une base de séquences de données (D').

Cette optimisation concerne la transformation de la base de données temporelle initiale (en entrée de l'algorithme) en une base de séquences de données qui n'est pas nécessaire dans notre algorithme basé sur la *BVPT,* à l'inverse de l'algorithme *PSP* où le calcul des supports passe inévitablement par la base de séquences de données et non pas par la base de données temporelle, ce qui contraint *PSP* avant toute analyse d'une base temporelle, d'opérer une telle transformation. Ceci nécessite à cet algorithme deux balayages coûteux en temps UC de cette base : le premier pour la trier par ordre croissant selon (*CID, DATE*) afin d'aboutir au second qui la réécrit en base de séquences de données.

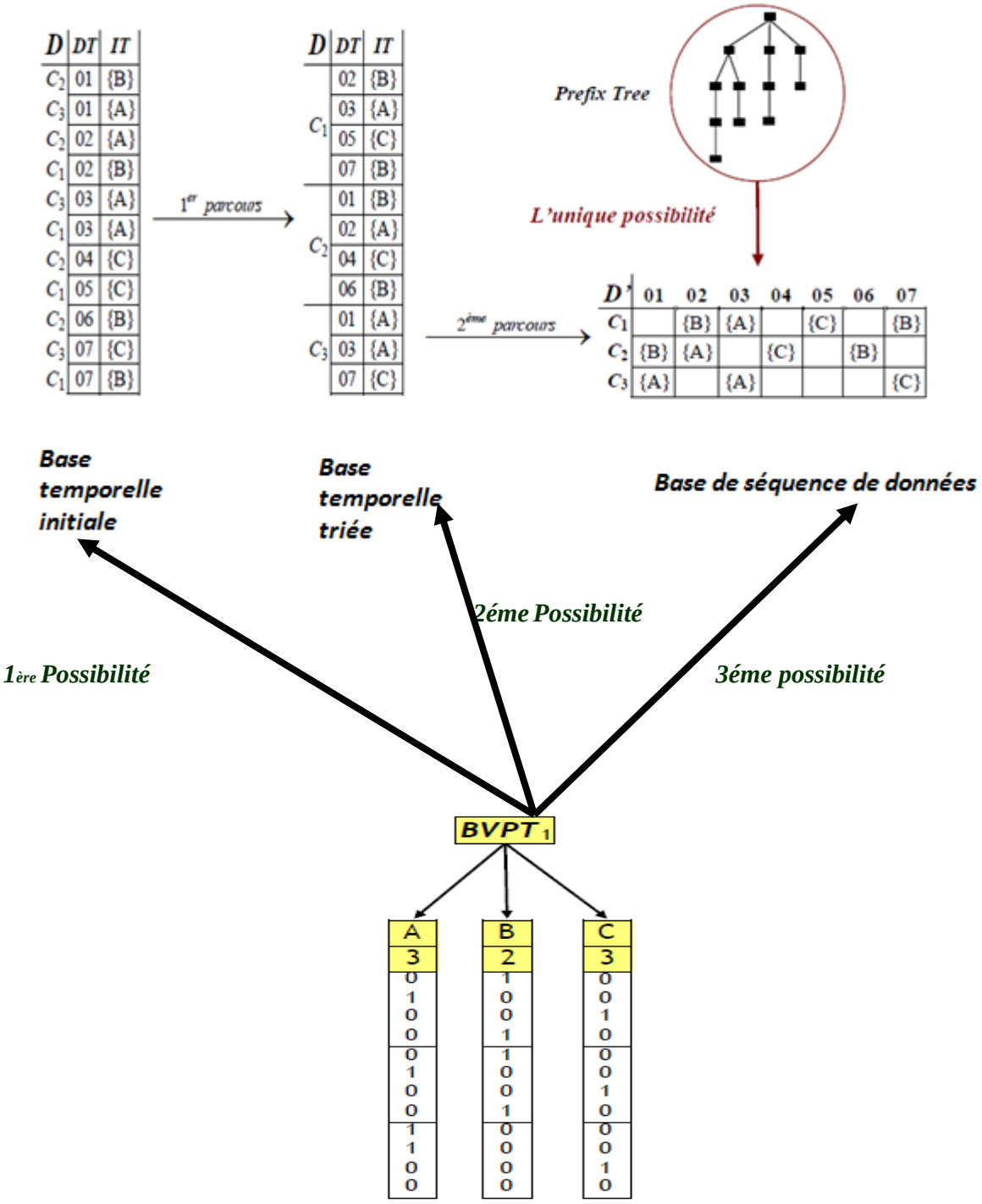


Figure 4.3 – Elimination de la phase de réécriture de la base de données par la structure BVPT

La *Figure 4.3* montre l'avantage de notre représentation du gain de deux parcours de la base initiale (*D*) tout en étant directement applicable sur une base de séquences de données (*D'*).Noter que cet avantage reste invariable pour toute autre méthode utilisant l'arbre *BVPT*.

4.2. Réduction du nombre de séquences candidates générées à chaque itération

Comme expliqué au chapitre précédent, l'algorithme *PSP* est contraint après chaque génération de séquence candidate de la représenter dans son arbre de préfixes avant de pouvoir calculer son support. Ceci s'explique par le fondement de cette méthode qui exige la projection de la base de séquences de données dans cet arbre afin d'incrémenter les supports de seulement les séquences candidates qui y sont représentées. Cette contrainte qui peut facilement générer une limitation mémoire dans le cas d'un nombre d'items fréquents très élevé, est éliminée dans notre méthode grâce à l'optimisation suivante.

Optimisation 3 : Elimination de l'étape de représentation dans l'arbre BVPT pour les séquences candidates pré identifiées non fréquentes (pré−élagage).

Nous proposons que les séquences candidates avant même d'être représentées dans l'arbre *BVPT*, qu'elles soient vérifiées fréquentes. Ceci peut paraître irréalisable, mais la *BVPT* grâce à sa conservation de l'information complète (en vecteurs de bits) sur les occurrences des séquences fréquentes identifiées à l'itération précédente (comme le fait *SPAM* dans sa recherche en profondeur mais, sur tous ses appels récursifs), permet de calculer le support d'une séquence candidate avant même d'étendre l'arbre par le dernier item ajouté à la séquence fréquente de niveau inférieur qui a contribué à sa génération. Ceci s'explique par le fait que, nous anticipons la génération d'une séquence candidate par le calcul du vecteur de bits lui correspondant, ce qui nous donne cet avantage par rapport à la méthode *PSP* ne possédant pas l'information nécessaire pour le réaliser.

Exemple

Considérons notre base de données temporelle *D* de la *Figure 4.3* et un seuil de support minimum de 2/3. Nous avons dans la *BVPT*1 (niveau 1) les vecteurs A et C représentant respectivement les occurrences des deux 1−séquences fréquentes <{A}> et <{C}> qui sont de support 3/3 dans *D*. Soit la 2−séquence candidate <{C}{A}> à générer. Notre démarche part du principe de ne représenter cette séquence dans la *BVPT*2 (niveau 2) que si elle est fréquente afin de minimiser ainsi, l'espace mémoire requis pour représenter l'ensemble des séquences candidates à considérer pour un niveau donné, ce qui est illustré comme suit.

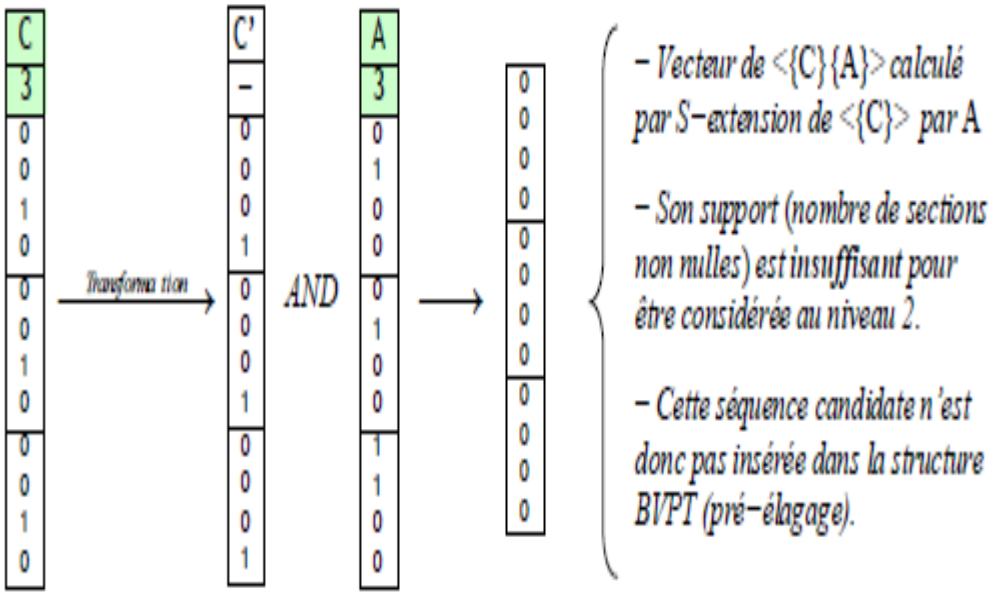


Figure 4.4 – Notre principe réduction du nombre de séquences candidates générées

4.3. Intégration de contraintes temporelles

Nous proposons dans cette section une extension très intéressante de l'algorithme *PSP*, permettant la prise en compte de contraintes temporelles (en plus de la contrainte de support minimum) dans les motifs séquentiels pendant la phase de recherche de ces motifs. Ceci est essentiel dans cette phase et ne peut être effectué ultérieurement ou en étape de posttraitement, du fait que les séquences fréquentes sous contraintes ou motifs séquentiels généralisés doivent de par leur définition, vérifier l'ensemble des paramètres temporels spécifiés par l'utilisateur pendant la phase de calcul de leurs supports et non pas après. Rappelons que les contraintes temporelles sur les motifs séquentiels sont au nombre de deux : L'intervalle de temps [*MinGap, MaxGap*] et la fenêtre temporelle *WinSize*.

Notre vision d'une telle dotation après l'intégration de notre nouvelle structure *BVPT* proposée, part du principe d'exploiter cette extension dans la vérification de ces deux paramètres pour chaque nouvelle séquence candidate générée, ce qui donne l'idée de base qui consiste à restreindre notre phase de sélection de séquences fréquentes à seulement celles vérifiant ces paramètres dans le calcul de leurs supports.

Deux solutions se sont alors présentées, représentant chacune une approche justifiée :

(1) La vérification de ces contraintes pendant la phase de calcul des vecteurs de bits relatifs aux séquences générées (par les *S−extensions* et les *I−extensions*), ce qui permet d'anticiper l'élimination des séquences non vérifiant ces paramètres à l'instar de l'extension *C−SPADE* de l'algorithme *SPADE* qui le fait pendant ses jointures temporelles.

(2) La vérification des contraintes temporelles pendant la phase de comptage des supports des séquences candidates générées après le calcul des vecteurs de bits leur correspondant.

Notre choix s'est fixé sur la solution (2) vu sa réalisabilité évidente et sa consommation réduite en temps UC malgré qu'en pratique, son implémentation dans la *BVPT* est complexe par rapport à d'autres méthodes classiques comme *GSP* [24]. Pour la solution (1), nous considérons que malgré qu'elle présente une meilleure performance, elle reste à notre connaissance, irréalisable dans le sens où le calcul des vecteurs de bits correspondant aux séquences candidates générées est immédiat dans les *S−extensions* et les *I−extensions,* indispensables pour notre génération de séquences candidates. Ceci est dû au fait que ce calcul est exclusivement basé sur l'opérateur "*ET*" logique, ce qui ne peut permettre en aucun cas d'intervenir en cours de cette opération indivisible.

En résultat, et partant du fait que notre *BVPT* tire sa partie représentation des occurrences des séquences candidates dans la base de données de la représentation bitmap de *SPAM*, nous proposons la modification suivante de la phase de calcul des supports de cette méthode, avant son adoption sous une nouvelle forme dans notre algorithme.

Extension 1 : Prise en compte de contraintes temporelles dans la phase de calcul des supports et son intégration dans l'algorithme PSP avec la nouvelle structure BVPT.

Afin de garantir la vérification de ces contraintes par les séquences candidates dans leurs inclusions dans les séquences de la base, nous proposons conjointement à l'optimisation 3(pour des raisons de minimisation de l'espace mémoire requis, comme le montre cette optimisation), d'étendre la phase de calcul des supports de ces séquences basée sur le comptage du nombre de sections non nulles dans les vecteurs de bits leur correspondant, à la vérification des distances temporelles *MinG*ap, *MaxGap* et *WinSize* dans ces vecteurs, ce qui apporte une modification majeure à cette phase consistant à ne plus compter le nombre de sections non nulles, mais, d'effectuer le comptage de la manière suivante :

(a) Chaque génération de vecteur de bits par *S−extension* déclenche pour le comptage de son support, la vérification de la contrainte d'intervalle de temps [*MinGap, MaxGap*] pour toutes les sections *CID* de ce vecteur afin de n'incrémenter le support de la séquence candidate qu'il représente que pour les sections où cette contrainte est vérifiée.

(b) Chaque génération de vecteur de bits par *I−extension* déclenche pour le comptage de son support, la vérification de la contrainte de fenêtre temporelle *WinSize* pour toutes les sections *CID* de ce vecteur afin de n'incrémenter le support de la séquence candidate qu'il représente que pour les sections où cette contrainte est vérifiée. De plus, et afin de corriger l'absence d'information sur les occurrences des items dans le vecteur résultat du fait que la *I−extension* ne peut supporter une transaction généralisée, nous apportons une modification importante àla manière par laquelle cette opération est réalisée.

Formellement, Soient $V(A) = ((a11, a12,\ldots, a1m)$; $(a21, a22,\ldots, a2m)$; … ; $(an1, an2,\ldots, anm))$ le vecteur de bits correspondent à l'item A de la base de données, $V(S) = ((s11, s12,\ldots, s1m)$; $(s21,s22,\ldots, s2m)$; … ; $(sn1, sn2,\ldots, snm))$ le vecteur de bits correspondant à une séquence fréquente *S*de niveau *k*, et $V(S')=((s'11, s'12,\ldots, s'1m)$; $(s'21, s'22,\ldots, s'2m)$; … ; $(s'n1, s'n2,\ldots, s'nm))$ le vecteur de bits correspondant à la séquence candidate *S'*de niveau $(k+1)$ générée par *S−extension* ou *I−extension* de la séquence *S* par l'item A, avec *n* le nombre de séquences de données dans cette base (ou nombre de *CID*) et *m* le nombre de bits par section (calculé en étape de projection de la base dans la *BVPT*, voir *Optimisation 1*).

(a) La vérification de la contrainte d'intervalle [*MinGap*, *MaxGap*] pour une section $i(i\in [1, n])$ du vecteur de bits $V(S')$ généré par *S−extension,* doit s'assurer qu'il existe $j,j'\in [1,m]$ tel que $V(S)[i,j] = 1$ et $V(S')[i,j'] = 1$ et $DATE(V(S')[i,j']) - DATE(V(S)[i,j]) \in [MinGap,MaxGap]$ avant d'incrémenter le support de *S'*pour cette section.

Exemple

Considérons notre base de données *D* de la *Figure 4.3* et le même seuil de support 2/3.

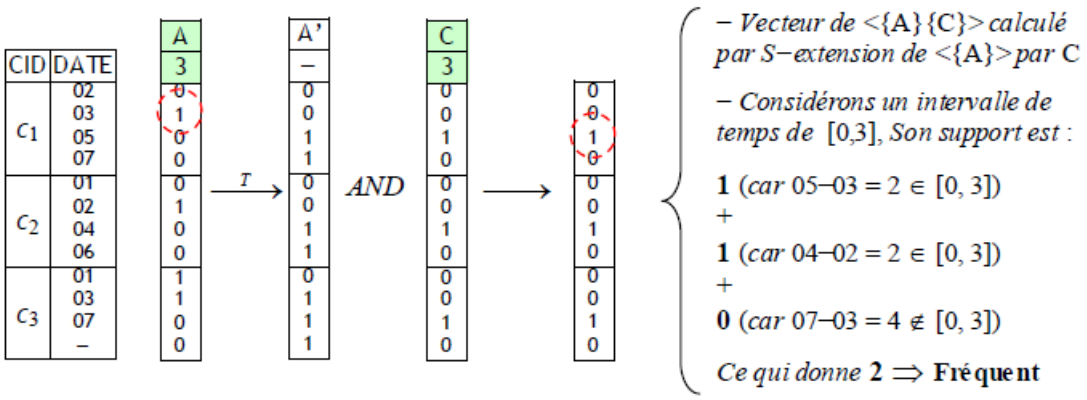


Figure 4.5 – Considération de la contrainte [MinGap, MaxGap] dans le calcul du support de la séquence < {A}{C}>

(b) La vérification de la contrainte de fenêtre temporelle *WinSize* pour une section $i (i \in [1,n])$ du vecteur $V(S')$ généré par *I–extension,* doit s'assurer qu'il existe $j,j' \in [1,m]$ tel que $V(S)[i,j] = 1$ et $V(A)[i,j'] = 1$ et $DATE(V(S)[i,j]) - DATE(V(A)[i,j']) < WinSize$, avant d'incrémenter le support de S' pour cette section. Nous modifions l'opération *I–extension* qui consistera à ne plus réaliser le *ET* logique entre $V(S)$ et $V(A)$ mais d'effectuer un *OU* sélectif visant à mettre $V(S')[i,j] = 1$ et $V(S')[i,j'] = 1$ pour chaque $(V(S)[i, j] = 1, V(A)[i, j'] = 1)$.

Exemple

Considérons notre base de données *D* de la *Figure 4.3* et le même seuil de support 2/3.

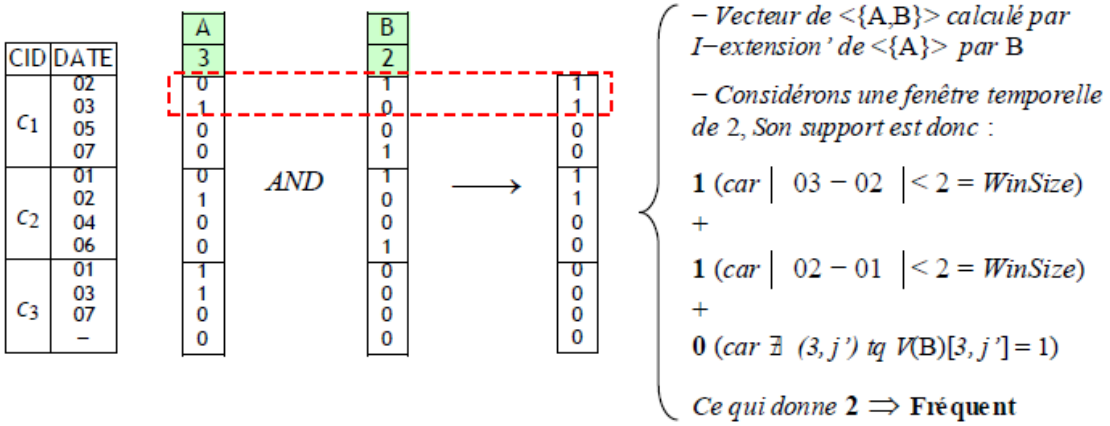


Figure 4.6 – Considération de la contrainte WinSize dans le calcul du support de la séquence <{A,B}>

Noter que la fonction *DATE* a été utilisée uniquement pour indiquer la valeur de l'attribut temporel (*DATE*) de la base de données pour les bits *V*(*S*)[*i j*]), *V*(*S'*)[*i,j'*]) et *V*(A)[*i,j*]) ayant la valeur 1 (occurrence d'item) dans les vecteurs *V*(*S*), *V*(*S'*) et *V*(A), qui sans cet identifiant distinctif, aucune recherche de motifs séquentiels ne peut être envisagée.

5. Algorithme d'extraction de motifs séquentiels sous contraintes proposé (PSBI)

Nous présentons dans cette section notre algorithme *PSBI* (*Prefix tree for Sequential pattern mining using BItmap représentation*) proposé pour l'extraction de motifs séquentiels sous contraintes temporelles dans une base de données. Comme expliqué dans la section précédente, il s'agit d'une méthode de recherche de ces motifs bâtie autour de la nouvelle structure de données *BVPT* que nous avons défini pour cet effet.

La BVPT (Bit VectorPrefixedTree) par sa représentation de la base de données en vecteurs de bits organisés dans un arbre de préfixes, permet non seulement de restreindre le nombre de parcours coûteux de cette dernière à l'unique lecture montrée dans la Figure 4.7, mais, nous lui avons doté la possibilité d'intégrer l'ensemble des paramètres temporels qui peuvent être spécifiés par l'utilisateur en plus du seuil de support minimum.

Après l'étape de préparation des données (BDD) et la définition des attributs d'extraction, ce qui aboutit à une base transactionnelle prête à l'analyse, l'algorithme *PSBI* est appliqué afin de réaliser son extraction. Dans cette phase, il commence par projeter cette base *transformée en vecteurs de bits verticaux* représentant les occurrences de ses items, dans l'arbre *BVPT* résident en mémoire centrale.

En appliquant les optimisations 1, 2, 3 et l'extension 1 relative à la composante contraintes temporelles, PSBI exploite cette structure par niveau, comme son unique espace de recherche de séquences fréquentes, dont il se sert pour calculer les occurrences ainsi que les supports des différentes séquences candidates générées pendant la recherche pour la construction de l'ensemble des séquences fréquentes découvertes parmi celles-ci, qui sont gardées dans cet arbre jusqu'à la fin de la recherche afin d'être retournées en résultat.

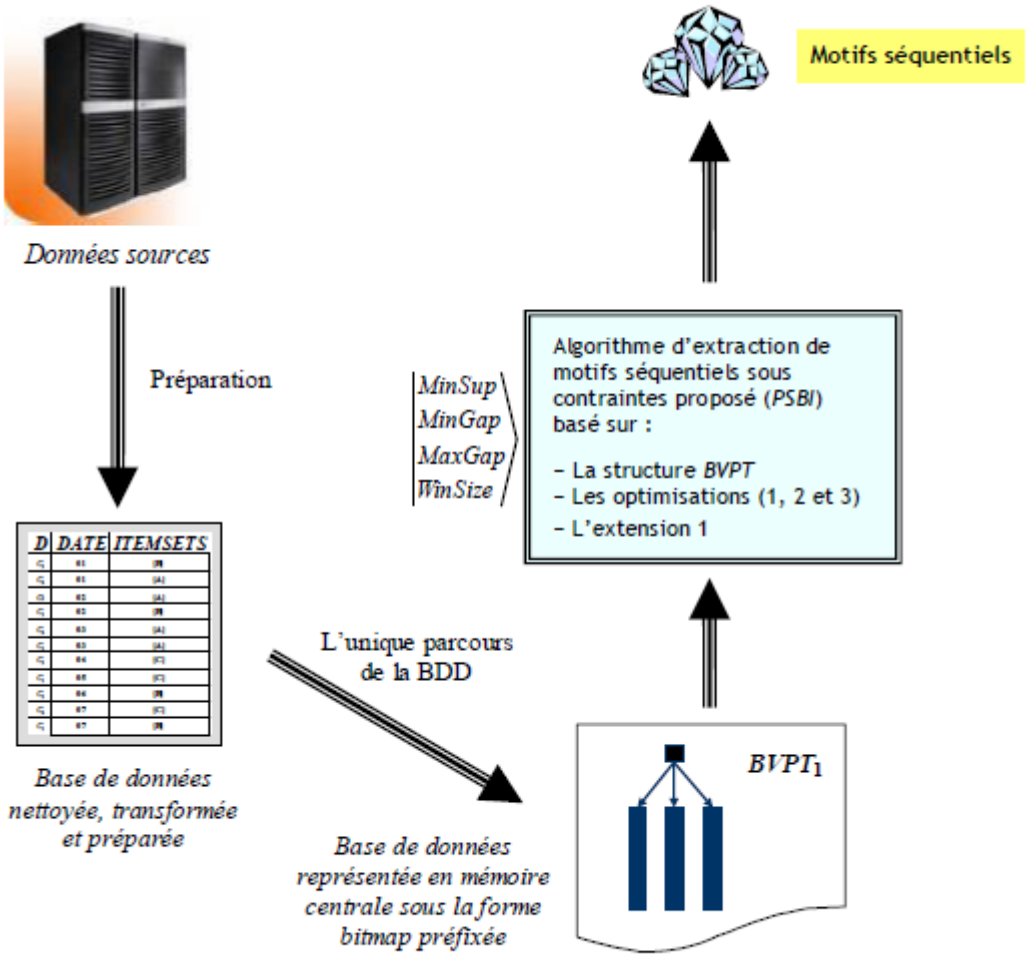


Figure 4.7 – Architecture générale de notre algorithme (PSBI)

Le pseudo-code suivant résume cette démarche, où la base de données en entrée de l'algorithme n'est parcourue qu'une unique fois pendant toute l'extraction grâce à l'optimisation 1. Quant au nombre de séquences générées, *PSBI* exploite fortement le principe de ne représenter une séquence candidate dans l'arbre *BVPT* que si elle est fréquente. Grâce à l'optimisation 3, le support d'une séquence candidate peut être connu à l'avance dans le vecteur de bits pré-calculé lui correspondant, comme le décrit cette optimisation.

Algorithme 4.1 : Algorithme d'extraction de motifs séquentiels généralisés proposé *(PSBI)* Extraction de motifs séquentiels sous contraintes {MinGap, MaxGap, WinSize} dans une base de données temporelle selon un seuil de support minimum MinSup.

Entrée : Base de données temporelle *D*;
Liste d'attributs I;
Support minimum *MinSup*;
Paramètres temporels : *MinGap*, *MaxGap* et *WinSize*
Sortie : Ensemble des séquences fréquentes dans *D*;
Construction de l'arbre *BVPT* (niveau 1) et y projection de *D* en appliquant l'optimisation 1
// La phase de transformation de *D* en base de séquences a été éliminée par l'optimisation 2

Pour chaque Transaction $T \in D$ **faire**
Pour chaque item $a \in ITEMSET(T)$ **faire**
Début
Si ∃/ noeud *N* de niveau 1 tel que *ITEM*(*N*) = *a* **alors**
Créer noeud $N \supset a$ fils *S−extension* de la racine
Vecteur(noeud ⊃ a)[*CID*(*T*), *DATE*(*T*)] = 1;
Fin;
// Extension des vecteurs de bits en *Nbre−CID* sections de *Nbre−bits−par−section* bits (par *CID*)
// avec la purge des noeuds de niveau 1 représentant des items non fréquents selon l'optimisation 1
Pour chaque nœud *N* de niveau 1 **faire**
Début
Etendre Vecteur(*N*) à *Nbre−bits−par−section* ∗ *Nbre−CID* bits;
// *Nbre−bits−par−section* et *Nbre−CID* définis lors de la projection de *D*
Si *Support−Item*(Vecteur(*N*)) < *MinSup* **alors** Supprimer *noeud N*;
Fin;
// Construction du niveau 2 : Génération, pré-élagage et représentation des 2−séquences fréquentes
// avec l'application de l'optimisation 3 : n'étendre un noeud que par une *S/I* –extension fréquente

Pour chaque Nœud N_1 de niveau 1 **faire**
Pour chaque Nœud N_2 de niveau 1 **faire**
Si *ITEM*(N_1) = *ITEM*(N_2) **alors**
Si *S−fréquente*(N_1, N_2) **alors**
Créer nœud $N' \supset ITEM(N_2)$ fils *S−extension* de N_1 avec Vecteur(*N*') = *S−extension*(N_1, N_2)
Sinon
Début
Si *S−fréquente*(N_1, N_2) **alors** Créer noeud $N' \supset ITEM(N_2)$ fils *S−extension* de N_1 avec Vecteur(*N*') = *S−extension*(N_1, N_2);
Si *I−fréquente*(N_1, N_2) **alors** Créer noeud $N' \supset ITEM(N_2)$ fils *I−extension* de N_1 avec Vecteur(*N*') = *I−extension'*(N_1, N_2);
Fin;
// Construction des niveaux supérieurs : Génération, pré-élagage et représentation des séquences
// fréquentes de niveau *i* avec l'application de l'optimisation 3 : extensions fréquentes uniquement

i= 2;

Répéter
Début
Pour chaque noeud *N* de niveau *i* **faire**
Début
// Traitement des noeuds fils par *S−extension*
Pour chaque noeud *N'* fils *S−extension* du nœud N_R fils de la racine contenant *ITEM*(*N*) **faire**
Si *S−fréquente*(*N*, N_R) **alors**
Créer noeud *N''* ⊃ *ITEM*(*N'*) fils *S−extension* de *N*
avec Vecteur(*N''*) = *S−extension*(*N*, N_R);
// Traitement des noeuds fils par *I−extension*
Pour chaque noeud *N'* fils *I−extension* du noeud N_R fils de la racine contenant *ITEM*(*N*) **faire**
Si *I−fréquente*(*N*, N_R) **alors**
Créer noeud *N''* ⊃ *ITEM*(*N'*) fils *I−extension* de *N*
avec Vecteur(*N''*) = *I−extension'*(*N*, N_R);
Fin; // non vérification du noeud frère (antimonotonicité) dans les motifs sous contraintes
Pour chaque noeud *N* de niveau *i* **faire** supprimer Vecteur(*N*); // Purge du niveau *i*
Incrémenter *i*;
Fin;
Jusqu'à (Aucun nœud *N''* de niveau *i* n'est crée par *S−extension* ou *I−extension*);
// Parcours récursif de l'arbre *BVPT* et construction de l'ensemble des séquences fréquentes
Retourner (Toutes les séquences maximales représentées dans l'arbre *BVPT* et leurs supports).

Fonction : *Support−Item* (*Vecteuri*)
Calcul du support de l'item représenté par le vecteur de bits *Vecteuri*

Nombre−sections−non−nulles = 0;
Pour *j*=1 **à** *Nbre−CID* **faire**
Si $SECTION_j(Vecteur_i)[1] \wedge SECTION_j(Vecteur_i)[2] \wedge \ldots$
$\ldots \wedge SECTION_j(Vecteur_i)[Nbre-bits-par-section] = 1$ **alors**
Incrémenter *Nombre−sections−non−nulles*;
Retourner *Nombre−sections−non−nulles.*

Fonction : *S−fréquente* (N_1, N_2)
Détermine si la *S−extension* de la séquence fréquente représentée par le nœud N_1, par l'item représenté par le noeud N_2 donne une séquence fréquente (Vérification de [*MinGap, MaxGap*]).

VS' = *S−extension*(N_1, N_2); // Calcul du vecteur de bits correspondant
Support = 0;
// Vérification de la contrainte [*MinGap, MaxGap*] pour chaque *CID* en se basant sur l'extension 1
Pour *i*=1 **à** *Nbre−CID* **faire**
Si $\exists\ j,j' \in [1, Nbre-bits-par-section]$ tel que $Vecteur(N_1)[i,j] = 1 \wedge VS'[i,j'] = 1 \wedge$
$DATE(VS'[i,j']) - DATE(Vecteur(N_1)[i,j]) \in [MinGap, MaxGap]$ **alors** incrémenter Support;

Si Support ≥ *MinSup* **alors Retourner** VRAI **Sinon Retourner** FAUX.

Fonction : *I−fréquente* (N_1, N_2)
Détermine si la *I−extension* de la séquence fréquente représentée par le noeud N_1, par l'item fréquent représenté par le nœud N_2 donne une séquence fréquente (Vérification de *WinSize*).

Support = 0;
// Vérification de la contrainte temporelle *WinSize* pour chaque *CID* en se basant sur l'extension 1
Pour *i*=1 **à** *Nbre−CID* **faire**
Si ∃ *j,j'* ∈ [1, *Nbre−bits−par−section*] tel que Vecteur(N_1)[*i,j*] = 1 ∧ Vecteur(N_2)[*i,j'*] = 1 ∧ *DATE*(Vecteur(N_1)[*i,j*]) − *DATE*(Vecteur(N_2)[*i,j'*]) < *WinSize*
alors incrémenter Support;
Si Support ≥ *MinSup* **alors Retourne r** VRAI **Sinon Retourner** FAUX.

Fonction : *S−extension* (N_1, N_2)
Calcule le vecteur de bits *S−extension* de la séquence fréquente représentée par le noeud N_1, par l'item fréquent représenté par le noeud N_2.

// Application de la *transformation SPAM* sur Vecteur(*N*1) décrite dans *Chapitre 3, Section 2.2.2.2*
Retourner (*Transformée* (Vecteur(N_1)) ∧ Vecteur(N_2)). // ET logique entre ces deux vecteurs

Fonction : *I−extension'* (N_1, N_2)
Calcule le vecteur de bits *I−extension modifiée* de la séquence fréquente représentée par le nœud N_1,par l'item fréquent représenté par le nœud N_2.

Pour *i*=1 **à** *Nbre−CID* **faire pour** *j*=1 à *Nbre−bits−par−section* **faire** *VI'*[*i,j*] = 0; // initialisation
Pour *i*=1 **à** *Nbre−CID* **faire**
Pour chaque *j,j'* ∈ [1, *Nbre−bits−par−section*] tel que Vecteur(N_1)[*i,j*] = 1 ∧ Vecteur(N_2)[*i,j'*] = 1 ∧ *DATE*(Vecteur(N_1)[*i,j*]) − *DATE*(Vecteur(N_2)[*i,j'*]) < *WinSize*
faire Début
VI'[*i,j*] = 1;
VI'[*i,j'*] = 1;
Fin;
Retourner *VI'*. // vecteur de bits calculé (*I−extension'*)

Application à un exemple :

Le présent exemple permet de mieux appréhender notre algorithme où nous proposons de l'appliquer notre base de données *D* introduite au début de ce chapitre, que nous reprenons sans transformation dans le contexte d'extraction donné par la *Figure 4.8*.

CID	*DATE*	*ITEMSET*
C_2	01	{B}
C_3	01	{A}
C_2	02	{A}
C_1	02	{B}
C_3	03	{A}
C_1	03	{A}
C_2	04	{C}
C_1	05	{C}
C_2	06	{B}
C_3	07	{C}
C_1	07	{B}

MinSup = 2	*MinGap* = 0
MaxGap = 3	*WinSize* = 2

Figure 4.8 – Contexte d'extraction pour la base temporelle D

Niveau 1 : Projection du contexte D dans l'arbre BVPT en mémoire centrale

Le contexte *D* est parcouru afin de construire le premier niveau de l'arbre *BVPT* en mémoire. Pour ceci, chaque item A, B et C de *D* est représenté par un nœud relié à la racine par une branche de type *S−extension*, ensuite, sont projetées les 11 transactions de *D* dans l'arbre ainsi construit où l'occurrence d'un item est représentée par 1 dans le vecteur de bits lui correspondant à la position (valeur *CID*, valeur *DATE*) de cette occurrence. A partir du maximum des nombres de ***valeurs distinctes de DATE*** **pour chaque *CID***, ce qui est pour cet exemple : *MAX* (4, 4, 3) = 4 pour 3 *CID*, *PSBI* partitionne les vecteurs de bits en 3 sections de4 bits chacun comme le montre la *Figure 4.9*. Ceci permet de calculer le support de chaque nœud (item), qui est le nombre de sections non nulles (contenant au moins un 1) dans le vecteur de cet item, pour ensuite ne garder dans la *BVPT* que les nœuds représentant un support supérieur ou égal à 2, c'est-à-dire : A(1+1+1 ≥ 2), B(1+1+0 ≥ 2) et C(1+1+1 ≥ 2).

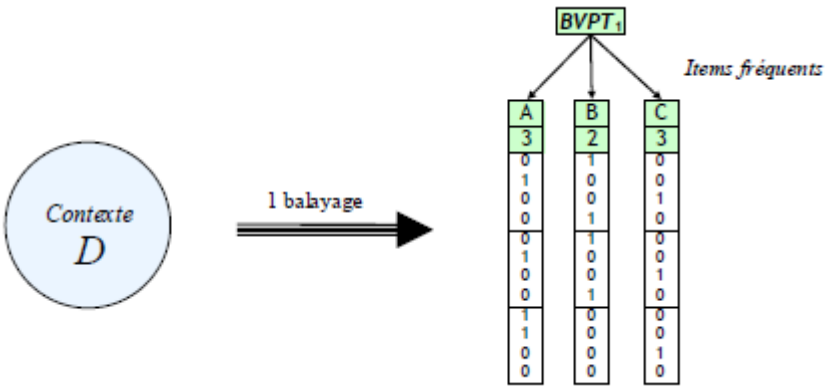


Figure 4.9 – Représentation bitmap préfixée (BVPT) du contexte D

Niveau 2 : Génération, pré−élagage et représentation des 2−séquences fréquentes

PSBI procède par une génération de 2−séquences candidates par la fusion (auto-jointure) deux à deux des 1−séquences fréquentes <{A}>, <{B}> et <{C}> représentées dans le niveau 1 de l'arbre *BVPT*, ce qui donne les 2−séquences candidates <{A}{A}>, <{A}{B}>, <{A}{C}>,<{B}{A}>, <{B}{B}>, <{B}{C}>, <{C}{A}>, <{C}{B}>, <{C}{C}>, <{A,B}>,<{A,C}> et <{C,B}>, où ne sont représentées (application de l'*optimisation 3*) dans le second niveau de cet arbre que celles ayant le support suffisant 2, calculé sur la base de la vérification des paramètres temporels [0,3] et 2 (application de l'extension 1).

▪ **Vérification de la contrainte d'intervalle de temps [0, 3]** : Les S−extensions des1−séquences fréquentes <{A}>, <{B}> et <{C}> chacune par les trois items A, B et C, calculent les vecteurs de bits donnés dans la Figure 4.10 ci-dessous, où chaque S−extension est suivie par la vérification de la contrainte d'intervalle [0, 3] dans les 3 sections du vecteur généré afin de calculer le support de la séquence qu'il représente, ce qui donne :

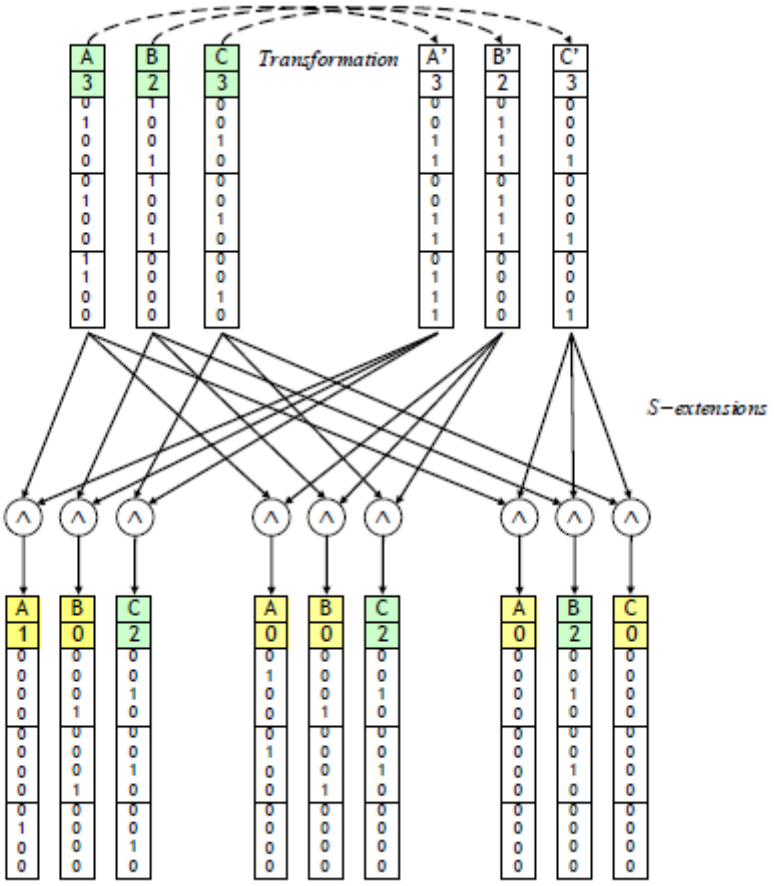


Figure 4.10 – Vecteurs de bits correspondant aux S−extensions de niveau 2

Les 9 vecteurs montrés dans la *Figure 4.10* correspondent aux 2−séquences candidates dans l'ordre : <{A}{A}>, <{A}{B}>, <{A}{C}>, <{B}{A}>, <{B}{B}>, <{B}{C}>,<{C}{A}>, <{C}{B}> et <{C}{C}>. Leurs supports ont été calculés suivant la méthode décrite dans l'extension 1 − vérification de la contrainte [*MinGap*, *MaxGap*]. Il en résulte que seules les séquences <{A}{C}>, <{B}{C}> et <{C}{B}> sont fréquentes. La manière par laquelle l'algorithme a effectué cette vérification a été illustrée dans la *Figure 4.5* montrée dans la *Section 3.3* pour l'exemple de la séquence <{A}{C}> dans le même contexte d'extraction (même base, même support et mêmes paramètres temporels).

▪ **Vérification de la contrainte de fenêtre temporelle *WinSize*= 2 :**Les I−extensions' de la1−séquence fréquente <{A}> par les deux items B et C et de la 1−séquence fréquente <{B}>par l'item C déclenchent la vérification de la contrainte de fenêtre temporelle WinSize= 2entre deux à deux, les trois sections des vecteurs de bits : A avec B, A avec C et B avec C, ce qui ne génère que le vecteur représentant la 2−séquence

fréquente <{A,B}> qui est supportée2 fois. La manière par laquelle l'algorithme a opéré a été illustrée dans la *Figure 4.6* montrée dans la *Section 3.3* pour l'exemple de cette séquence.

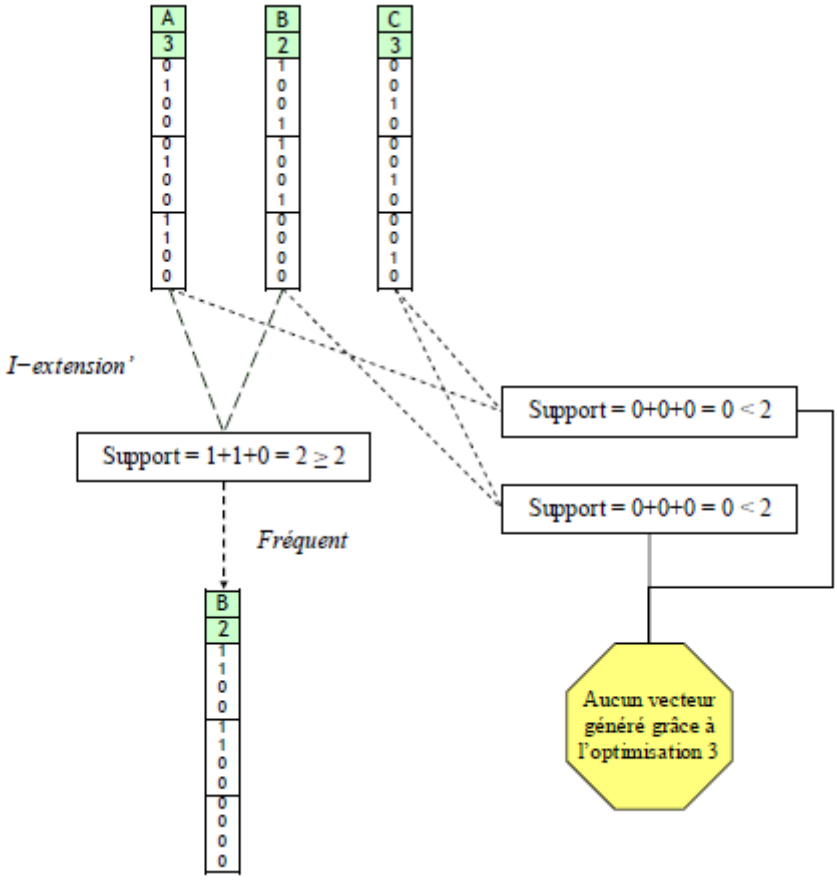


Figure 4.11 – Vecteur de bits correspondant aux I−extensions' de niveau 2

En somme, l'algorithme aurait identifié pour ce niveau les quatre 2−séquences fréquentes :<{A}{C}>:2 , <{B}{C}>:2, <{C}{B}>:2 et <{A,B}>:2. Noter qu'uniquement celles-ci (et les vecteurs de bits leur correspondant) sont représentées dans l'arbre *BVPT* niveau 2 et non pas toutes les 12 séquences candidates considérées, comme le montre la figure suivante.

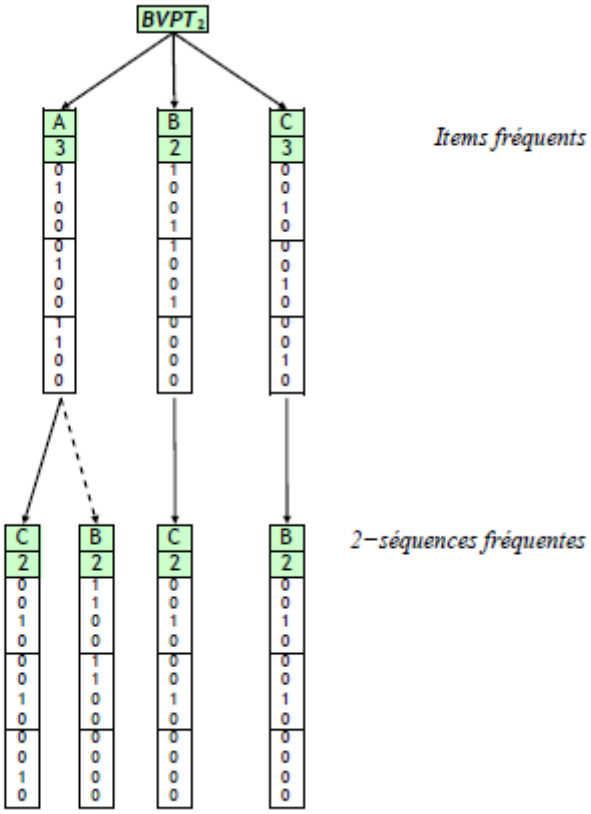


Figure 4.12 – Extension de l'arbre BVPT parles 2−séquences fréquentes uniquement

Niveau 3 : Génération, pré−élagage et représentation des 3−séquences fréquentes

La génération de séquences candidates pour le niveau 3 et supérieur ne nécessite aucune fusion de séquences puisqu'elle tire profit de la structure préfixée de l'arbre *BVPT*. Pour chacun des nœuds de second niveau C, B, C et B, l'algorithme cherche les fils de ce nœud à partir de la racine afin de ne l'étendre par la suite que par les items représentés par ceux-ci, ce qui permet de générer toutes les 3−séquences candidates préfixées par la 2−séquence fréquente représentée par ce nœud. Remarquer qu'aucune vérification de nœuds frères n'est effectuée dans cette génération. Ceci est dû au fait que la propriété d'**antimonotonicité** n'est pas vérifiée dans le cas des motifs sous contraintes. Détaillons dans ce qui suit les extensions des deux nœuds fréquents : C fils de A et B fils de C.

− **Nœud C fils de A** : l'algorithme cherche le nœud C à la racine et trouve son fils unique, le nœud B. Le nœud C est donc étendu par l'item B à la condition de vérifier le paramètre d'intervalle [0,3] dans le support de cette *S−extension*, ce qui est une application de l'extension 1 et l'optimisation 3 combinées. Le calcul du vecteur de bits correspondant montré dans la *Figure 4.13*, donne un support égal à 2, ce qui permet de réaliser cette *S−extension* et de représenter ainsi la séquence fréquente <{A}{C}{B}> dans le niveau 3 de l'arbre *BVPT*, comme le montre également cette figure.

− **Nœud B fils de C** : l'algorithme cherche le nœud B à la racine et trouve son fils unique, le nœud C. Le calcul de la *S−extension* du nœud B fils de C par l'item C donne un vecteur débit nul. Aucune extension du nœud B n'est donc effectuée, ce qui permet de réaliser un gain non négligeable en espace mémoire sur le vecteur de bits lui correspondant.

En somme, *PSBI* aurait découvert pour ce niveau les trois 3−séquences fréquentes :<{A}{C}{B}>:2 , <{A,B}{C}>:2 et <{B}{C}{B}>:2, représentées comme suit.

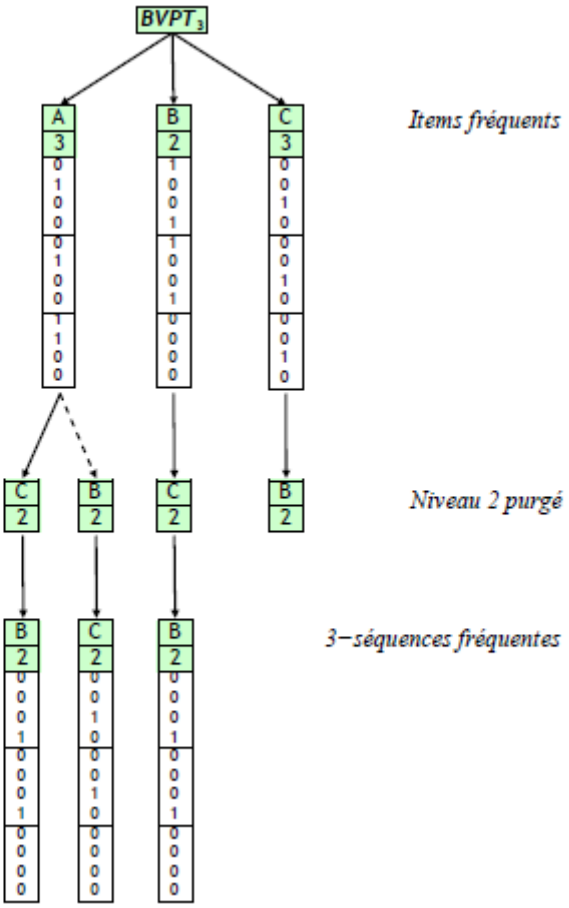


Figure 4.13 – Niveau 3 de l'arbre BVPT avec la purge du niveau 2

Niveau 4 : Génération, pré−élagage et représentation des 4−séquences fréquentes

Pour chacun des nœuds de troisième niveau B, C et B, *PSBI* cherche les fils de ce nœud à partir de la racine afin de ne l'étendre que par ceux parmi ceux-ci qui donnent des *S−extensions* fréquentes. Les *S−extensions* des deux nœuds B fils des deux nœuds C par l'item C (fils de B de la racine), donnent deux vecteurs nuls. Il en reste la seule *S−extension* du nœud C fils de B par l'item B (fils de C de la racine), qui donne un vecteur 2−fréquent après la vérification de la contrainte [0,4] dans chacune de ses 3 sections. L'arbre *BVPT* de niveau 4 obtenu après la purge des vecteurs de niveau 3 (non réutilisables) est comme suit.

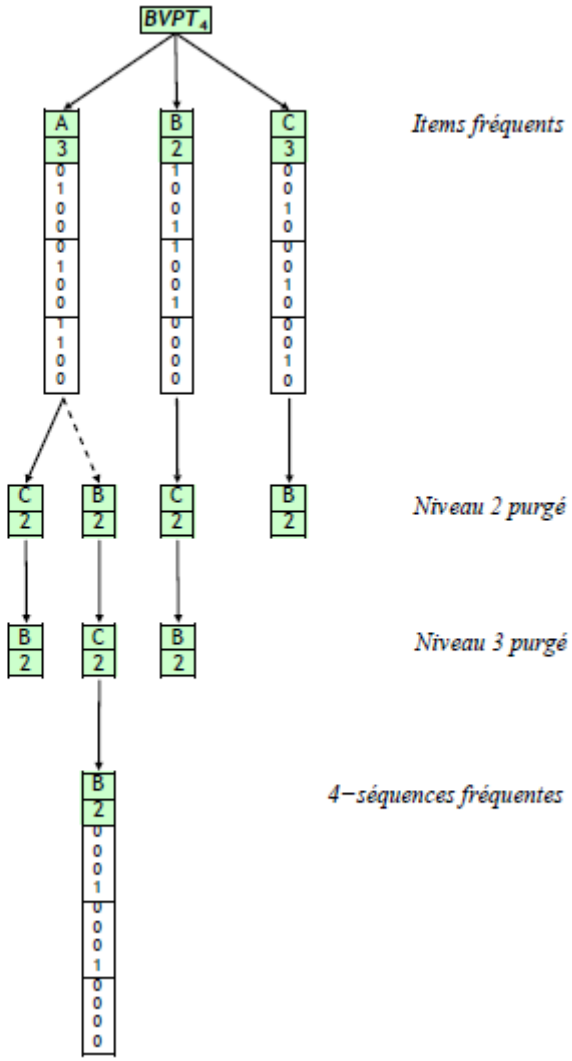


Figure 4.14 – Niveau 4 de l'arbre BVPT avec la purge du niveau 3

Niveau 5 : Aucune 5−séquence fréquente identifiée

La seule *S−extension* possible du nœud B fils de C par l'item C fils de B de la racine, donne un vecteur de bits nul, ce qui ne permet de réaliser aucune extension du niveau 4 de l'arbre. La condition d'arrêt est donc vérifiée.

L'algorithme arrête sa recherche et retourne en résultat l'ensemble des 11 séquences trouvées au total avec leurs supports, autrement dit, toutes celles représentées dans la dernière instance de l'arbre *BVPT* donné par la *Figure 4.14*, c'est-à-dire : <{A}>:3, <{B}>:2, <{C}>:3,<{A}{C}>:2, <{A,B}>:2, <{B}{C}>:2, <{C}{B}>:2, <{A}{C}{B}>:2, <{A,B}{C}>:2,<{B}{C}{B}>:2, et <{A,B}{C}{B}>:2.

Quant aux séquences fréquentes maximales, elles sont facilement dérivées de cette même structure grâce à sa représentation particulièrement efficace de toutes les inclusions dans cet ensemble, ce qui donne : <{A,B}{C}{B}>:2.

Résumons :

On peut dire que la recherche de motifs séquentiels généralisés se fait dans notre algorithme à partir de la base de données temporelle représentée en mémoire centrale sous une forme bitmap préfixée (*BVPT*) très utile, ce qui le rend plus performant en terme de temps de réponse grâce à son unique parcours de cette dernière et aux opérations binaires très efficaces qu'offre cette représentation. Quant à l'utilisation des ressources mémoire, notre algorithme veille après le traitement de chaque niveau, à libérer l'espace mémoire occupé par les vecteurs de bits du niveau directement inférieur. Ces derniers qui ne sont plus nécessaires à l'avancement de sa recherche, sont donc **purgés** de la structure *BVPT* (*Figures 4.13* et *4.14* de l'exemple). De plus, l'algorithme est directement applicable sur la base temporelle (ce qui est très souvent le cas en pratique), comme il peut facilement s'adapter à une base de séquences de données, par une très simple modification de sa phase initiale de projection.

6. Illustration de l'algorithme PSBI sur la BDD de la maintenance industrielle

6.1. Motivations et opportunités

Partant de l'étude effectuée tout au long de ce mémoire, nous proposons l'application de la technique des motifs séquentiels généralisés à la base de données de ***maintenance***

***industrielle**, dans* l'objectif d'identifier des relations temporelles entre pannes et défauts des équipements et installations industrielles, sous forme de séquences de pannes intéressant les spécialistes en maintenance. Notre visée principale est de permettre à ces derniers d'agir à l'avance sur des types de pannes estimées très coûteuses ou à risque sur la sécurité, ce qui permettra de réduire énormément les coûts de la maintenance et de sauver des vies humaines dans le cas extrême. Il s'agit d'un mode de maintenance plus que préventif, c'est la **maintenance industrielle anticipée**.

- La table utilisée concerne l'historique des demandes de travaux de maintenance sur les équipements.

Définition de la base transactionnelle (contexte d'extraction)

Il s'agit de choisir les trois attributs clés sur la base desquels sera effectuée l'analyse. En effet, nous savons que toute recherche de motifs séquentiels est fondamentalement basée sur : ***CID*** qui représente l'identifiant des séquences de données, ***DATE*** qui définit l'identifiant temporel de la base, et ***ITEMSET*** qui donne l'ensemble des items d'une transaction donnée pour un *CID* donné à une *DATE* donnée. De là, on peut tirer plusieurs analyses selon le choix de ces trois attributs importants :

(a) Découvrir des enchaînements de pannes fréquents sur les mêmes équipements (séquences de pannes fréquentes **par équipement**). Dans ce cas, l'attribut *CID* est l'équipement, l'attribut *DATE* est la date d'établissement et l'attribut *ITEMSET* est l'article.

(b) Découvrir des enchaînements de pannes fréquents quelque soit l'équipement (séquences de pannes fréquentes). Dans ce cas, l'attribut *CID* est le numéro de la demande de travail, l'attribut *DATE* est la date d'établissement et l'attribut *ITEMSET* est l'article.

(c) Découvrir des enchaînements fréquents d'équipements en panne (séquences fréquentes d'équipements en panne). Dans ce cas, l'attribut *CID* est l'article, l'attribut *DATE* est la date d'établissement et l'attribut *ITEMSET* est l'équipement.

notre choix s'est défini par la première analyse (a), du fait que les spécialistes en maintenance cherchent souvent à comprendre le comportement dans le temps des différentes pannes que subisse un équipement donné, notamment lorsqu'il s'agit d'un équipement stratégique pour la production dont les temps d'arrêt sont très coûteux ou d'un équipement pouvant représenter une séquence de pannes particulière, aboutissant par exemple, à une défaillance à risque sur la sécurité.

Le format de la base de transactions correspondante, la table ***MAINT_EQUI*** est le suivant :

1/La base temporelle initiale triée selon ID temporel :

ID de la séquence (ID équipement)	**ID temporel (Date établissement)**	**Article (itemset)**
Eq2	04/02/2014	A2
Eq3	04/02/2014	A1
Eq2	30/03/2014	A1
Eq1	13/04/2014	A2
Eq3	13/04/2014	A1
Eq1	10/01/2015	A1
Eq2	20/02/2015	A3
Eq1	13/03/2015	A3
Eq2	15/05/2015	A2
Eq3	13/08/2015	A3
Eq1	13/09/2015	A2

Avec :

-Eq1 : condenseur de propane

-Eq2 : Refroidisseur

-Eq3 : pompe d'alimentation

Et :

-A1 : purgeur automatique série 300

-A2 : Courroie trapézoïdale en nappe

-A3 : joint circulaire

2/La base temporelle triée selon ID équipement + ID temporelle :

D	Date transaction	itemset
Eq1	13/04/2014 10/01/2015 13/03/2015 13/09/2015	A2 A1 A3 A2
Eq2	4/02/2014 30/03/2014 20/02/2015 15/05/2015	A2 A1 A3 A2
Eq3	4/02/2014 13/04/2014 13/08/2015 –	A1 A1 A3 –

6.2.Recherche de séquences de pannes fréquentes

Le tableau suivant présente quelques résultats (séquences de pannes fréquentes par équipement) obtenus à partir de la table *MAINT_EQUI* (*CID* = l'équipement, *DATE* = date d'établissement et *ITEMSET* = l'article), selon différentes valeurs du seuil de support minimum, en considérant le même ensemble de paramètres temporels :

− **Intervalle de temps : [*MinGap*= 0, *MaxGap*= 365] jours.**

− **Fenêtre temporelle : *WinSize*= 6 mois.**

Support minimum	< Séquences de pannes fréquentes > : support
3%	< {A1}> :3, < {A3}> :3
2%	< {A1}> :3, < {A2}> :2,<{A3}> :3,<{A1}{A3}> :2, <{A2}{A1} > :2,<{A2}{A2} > :2,<{A3}{A2 }> :2 , < {A2A3} > :2, < {A1}{A3}{A2}> :2, < {A2}{A1}{A3}> :2, <{A2A3}{A2}> :2, < {A2}{A1}{A3}{A2}> :2
1%	< {A1}> :3, < {A2}> :2,<{A3}> :3, <{A1}{A1} > :1,<{A1}{A2} > :1,<{A1}{A3} > :2,<{A2}{A1} > :2, <{A2}{A2} > :2,<{A2}{A3}> :1,<{A3}{A2}> :2,< {A1A2} > :1, <{A2A3} > :2, <{A1A2}{A1}> :1 , <{A1A2}{A3}> :1 , <{A1}{A2A3}> :2 , <{A1}{A3} {A2}> :2 , <{A2}{A1} {A2}> :1 , <{A2}{A1} {A3}> :2, <{A2}{A1A2}> :1, <{A2}{A2A3}> :2, <{A2}{A3} {A2}> :2, <{A2A3} {A2}> :2, <{A3}{A2A3}> :2, <{A1A2}{A1A2}> :1, <{A1A2}{A1}{A 3}> :1, <{A1A2}{A3}{A2}> :1, <{A1}{A2A3} {A2}> :1, <{A1}{A3}{A2A3}> :2, <{A2}{A1}{A2A3}> :2, <{A2}{A1}{A3}{A2}> :2, <{A2}{A1}{A2A2}> :2, <{A2}{A2A3}{A2}> :2 , <{A2}{A3}{A2A3}> :2, <{A2A3}{A2A3}> :2, <{A1}{A2A3}{A2}> :2

Tableau 4.1 – Séquences de pannes fréquentes à partir de la table MAINT_EQUI selon différents supports

Afin de montrer l'effet des paramètres temporels sur l'extraction, le tableau suivant présente quelques motifs séquentiels obtenus à partir de la même table *MAINT_EQUI* selon un seuil de support minimum de 2 % avec une variation significative de ces paramètres.

Win	min	max	<Séquences de pannes fréquentes >: support

size	gap	gap	
1mo si	0	5mo i	**<{A1}> :3,<{A2}> :2,<{A3}> :3,<{A2}{A2}> :2,**
2mo i	0	6mo i	**<{A1}> :3,<{A2}> :2,<{A3}> :3,<{A2}{A2}> :2,<{A3}{A2}> :2**
4mo i	0	9mo i	**<{A1}> :3,<{A2}> :2,<{A3}> :3,<{A2}{A1}> :2,<{A2}{A2}> :2 ,<{A3}{A2}> :2.**

Tableau 4.2 – Séquences de pannes fréquentes à partir de MAINT_EQUI selon différentes valeurs des paramètres temporels

6.3. Signification de quelques motifs intéressants

A partir du *Tableau 4.1*, expliquons quelques règles séquentielles intéressantes qui ont pu être découvertes selon un support minimum de 2 % avec la considération d'un intervalle de temps de [0, 365] jours et d'une fenêtre temporelle de 6moi.

La règle séquentielle 01 :<{A1}→ {A3}>

SI

A1 : purgeur automatique série 300

ALORS dans un délai maximum de (365 jours)

A 3 : joint circulaire

La règle séquentielle 02:< {A2A3} >

Les deux remplacements suivants sont effectués dans les même 6mois:

A2 : Courroie trapézoïdale en nappe

ET

A3 :joint circulaire

La règle séquentielle 03 :< {A2A3}→{A2}>

SI

Les deux remplacements suivants sont effectués dans les même 6mois:

A2 : Courroie trapézoïdale en nappe

A3 :joint circulaire

ALORS dans un délai maximum de (365 jours)

A2 : Courroie trapézoïdale en nappe

La règle séquentielle 3 :< {A2} → {A1}→{A3}>

SI

A2 : Courroie trapézoïdale en nappe

ALORS dans un délai maximum de (365 jours)

A1 :purgeur automatique série 300

ET

SI

{A2}→ {A1}est vérifié

ALORS dans un délai maximum de (365 jours)

A3 :joint circulaire

Explication

-La règle séquentielle 1 exprime que si l'on remplace le purgeur automatique série 300 de l'équipement condenseur de propane on aura à remplacer dans un délai d'une année maximum le joint circulaire de la pompe d'alimentation.

-La règle séquentielle 2 montre que les remplacements de Courroie trapézoïdale en nappe du Refroidisseur et le joint circulaire série 300 de la pompe d'alimentation sont effectué fréquemment dans les même 6 mois.

-La règle séquentielle 3 montre que si les remplacements de Courroie trapézoïdale en nappe du Refroidisseur et le joint circulaire série 300 de la pompe d'alimentation sont effectué fréquemment dans les même 6 mois ALORS on aura à remplacer dans un délai d'une année maximum la Courroie trapézoïdale en nappe de ce refroidisseur.

-La règle séquentielle 4 montre que si l'on remplace la courroie trapézoïdale en nappe du refroidisseur ALORS on aura a remplacer dans un délai maximum d'une année le purgeur automatique série 300 du condenseur de propane et ce qui implique le remplacement dans un délai maximum d'une année du joint circulaire de la pompe d'alimentation.

7. Comparaison de l'algorithme PSP et PSBI et Résultats

De manière à analyser les performances de notre algorithme, nous avons effectué différentes expérimentations basées sur des données réelles (de maintenance) avec des variations significatives des paramètres du contexte : seuil de support minimum et paramètres temporels. Nous allons en premier lieu comparer les résultats de notre algorithme avec ceux de l'algorithme *PSP* pour démontrer qu'il est plus performant en termes de temps de réponse avec une utilisation raisonnable de l'espace mémoire. Ensuite, nous montrons l'apport de la recherche sous contraintes qu'adopte notre algorithme, dans la réduction du nombre de règles non intéressantes.

7.1. Réduction du nombre de parcours de la base de données

le tableau suivant montre le nombre de parcours effectués sur la base transactionnelle *MAINT_EQUI* par notre algorithme ainsi que par l'algorithme *PSP*, selon différentes valeurs du seuil de support minimum.

Support Minimum	**Longueur maximale des motifs**	**Nombre de parcours de la base transactionnelle**

	extraits	PSP	PSBI
1%	4	6	**1**
2%	4	6	
3%	1	3	

Tableau 4.3 – Nombre de parcours de la base transactionnelle

Selon différentes valeurs du support minimum

Nous savons que le nombre de parcours de la base transactionnelle par l'algorithme *PSP* est égal à longueur maximale des motifs extraits plus les deux parcours effectués en phase de transformation de cette base en base de séquences de données. Le *Tableau 4.3* montre que la longueur des motifs extraits augmente en proportion inverse du seuil de support minimum, ce qui implique une consommation importante de temps UC dans des opérations d'entrée/sortie dans le cas d'une base de données très dense même de taille raisonnable. Ce problème est résolu dans notre algorithme grâce à sa représentation des occurrences des items de cette base dans l'arbre *BVPT*, ce qui permet le gain d'un parcours par niveau. De plus, aucune transformation en base de séquences de données n'est effectuée du fait qu'il est directement applicable sur la base transactionnelle initiale. C'est pour ces raisons que le nombre de parcours de la base de données par notre algorithme est, quel que soit le seuil de support minimum, réduit à l'unique lecture montrée dans le *Tableau 4.3*.

7.2. Réduction du nombre de séquences candidates générées

En se basant sur les paramètres temporels de la section précédente, le tableau suivant montre le nombre de nœuds générés dans les deux structures : *PrefixTree* et *BVPT*, représentant l'ensemble des séquences considérées par notre algorithme ainsi que par l'algorithme *PSP*, selon différentes valeurs du seuil de support minimum

Support	Nombre de nœuds

Minimum	Prefixtree	BVPT
1%	33	35
2%	31	12
3%	10	2

Tableau 4.4 – Nombre de nœuds générés selon différents supports

Le tableau 4.4 montre le nombre de nœuds entre les deux structures. On voit bien que pour la structure *BVPT*, et aussi pour préfix tree le nombre de nœuds générés croît faiblement par rapport à des variations décroissantes importantes du seuil de support minimum, et a chaque fois que le seuil du support minimum augmente on aura le nombre de nœud généré décroit. Pour la structure BVPT Ceci s'explique par la stratégie de *pré-élagage* adoptée par notre algorithme qui n'étend un niveau donné de l'arbre *BVPT* que par des nœuds représentant des séquences réellement fréquentes, ce qui représente un gain très important de temps UC ainsi qu'une optimisation intéressante de l'espace mémoire requis pour ces traitements

7.3. Utilisation de l'espace mémoire

La structure *BVPT* nécessite une occupation mémoire plus élevée que la structure *PrefixTree*. En effet, malgré la réduction du nombre de nœuds et l'application de l'optimisation qui consiste à purger les vecteurs de bits non réutilisables pour les niveaux supérieurs, l'occupation mémoire de notre algorithme reste plus importante que celle de *PSP*. Toutefois, nous ne considérons pas ceci comme une grande limitation, car cet espace outre le fait qu'il a tendance à se limiter pour des variations décroissantes importantes du seuil de support minimum (ce qui est intéressant pour les bases de données denses), s'explique par fait de la préservation l'arbre *BVPT* dans son premier niveau *uniquement*, de l'information sur les occurrences des items fréquents, indispensable, non seulement au calcul des supports sans recours à des parcours coûteux de la base de données, mais surtout à la vérification des contraintes

temporelles. Ces dernières sont, comme nous l'avons montré dans ce mémoire, non prises en charge par l'algorithme *PSP*.

En somme, l'apport de la recherche de motifs généralisés qu'apporte notre algorithme, dans la réduction du nombre de règles non intéressantes, est une caractéristique importante vu sa temporelle dûment demandée par les applications exigeantes

8. Conclusion

Dans ce chapitre, nous avons présenté notre approche d'extraction de motifs séquentiels dans une base de données temporelle. Une démarche expérimentale était nécessaire pour mener à bien notre processus de découverte de connaissances. La première étape qui concerne la préparation de la base de transaction.

Dans la deuxième étape, qui concerne la recherche proprement dite de ces motifs, nous avons appliqué notre algorithme *PSBI* avec sa nouvelle structure de données hybride proposée, l'arbre *BVPT*, qui permet un comptage de supports efficace basé dans tous ses niveaux sur l'unique parcours de la base transactionnelle pendant la phase initiale de sa projection.

Toutefois, notre contribution principale dans cette structure réside dans le fait que nous l'étendons à la prise compte de contraintes temporelles qui peuvent être spécifiées par l'utilisateur en plus du seuil de support minimum.

Les différentes expérimentations effectuées suivant des variations significatives de ces paramètres (support minimum et paramètres temporels) montrent l'intérêt de la recherche sous contraintes qu'adopte notre algorithme, qui génère en sortie un nombre réduit de motifs séquentiels intéressants. Ceci s'ajoute à l'intérêt de l'arbre *BVPT* qui peut être adaptée à tout algorithme basé sur une représentation préfixée des séquences.

Ainsi, les résultats élaborés sur les motifs extraits et leur pertinence, peuvent aider les opérateurs en maintenance à définir des contrôles supplémentaires et des actions d'entretient basées sur l'anticipation des pannes coûteuses ou à risque sur la sécurité, ce qui peut s'ajouter comme élément complémentaire très utile à la fonction de maintenance.

Conclusion générale

Après l'engouement des travaux de recherche pour les règles d'association, les motifs séquentiels ont été très étudiés ces dernières années. Nous avons pu constater au cours de ce mémoire que depuis la définition de la problématique en 1995, de nombreux travaux de recherche ont été menés. Initialement, les premiers travaux ont consisté à améliorer les performances des propositions. Dans ce cadre, de nouvelles structures de données ou de nouvelles représentations des données sources ont été mises au point.

D'autres extensions ont considéré qu'avec les nouvelles techniques de stockage, il devenait possible de gérer la base en mémoire vive et proposent ainsi de nouveaux algorithmes. Même si les propositions ont permis de considérablement améliorer les temps de réponse, un premier constat a émergé. De la même manière que pour les règles d'association, il n'est pas possible de dire quel algorithme est meilleur que les autres, dans la mesure où leurs performances sont étroitement liées aux types de données manipulées. Ainsi, les algorithmes du type GSP ou PSP seront très efficaces dans le cas de grandes bases de données avec des séquences moyennement longues.

Par contre, les algorithmes comme SPADE, SPAM ou PREFIXSPAN seront eux très efficaces dans le cas où un très grand nombre de candidats de même taille sont générés. L'utilisation des algorithmes sur différents domaines d'application et le fait que les données sources peuvent varier rapidement ont donné lieu à de nombreux travaux de recherche autour de la fouille de données incrémentale.

L'une des raisons essentielles du développement des approches autour des motifs séquentiels est leur capacité d'adaptation à de très nombreux problèmes. Pour faciliter cette adaptation, les derniers travaux intègrent d'ailleurs de plus en plus de contraintes et offrent plus de souplesse sur la définition des motifs séquentiels. Parmi les nouveaux problèmes utilisant des techniques directes ou issues des motifs séquentiels nous pouvons Citer :

– le *Web Mining*[33] qui consiste à analyser les sites Web en fonction de leurs structures et de leurs contenus (Web Content Mining) mais également en fonction de leurs usages (Web Usage Mining).

– le *Schéma mining*[34] qui consiste à rechercher dans de grandes bases de données d'objets complexes des structures typiques pour par exemple créer de manière automatique des index ou des vues.

– le *Text Mining* dans lequel les motifs séquentiels peuvent servir à analyser des tendances au cours du temps dans des textes [35],

– et bien d'autres problèmes relatifs à la bio-informatiques (séquences de gènes, relevés de capteurs bio-médicaux, ...) [36], à l'accidentologie, la sécurité (détection d'intrusions, détection de fraudes), etc.

Perspectives

Les travaux présentés dans ce mémoire offrent de nombreuses perspectives de recherche ultérieure tant au niveau théorique qu'au niveau pratique. Dans cette section, nous présentons succinctement quelques-unes des perspectives qui nous paraissent être intéressantes.

Perspectives fonctionnelles

− Comme on a pu le mentionner dans le chapitre 2 de ce mémoire, le traitement de données séquentielles concerne d'innombrables domaines d'application. Parmi ceux-ci, l'analyse des séquences *ADN* qui en est un très actif ayant déjà donné lieu à de nombreux travaux.

Malheureusement, la majorité des techniques développées sont très spécifiques au domaine et très peu d'entre elles exploitent la technique des motifs séquentiels. Il serait donc intéressant d'étudier l'intérêt que pourrait avoir cette technique sur ce domaine particulier, notamment en cherchant à intégrer la prise en compte de contraintes temporelles, qui pourraient avoir dans ce cas, une signification différente que celle que nous avons connu dans ce mémoire.

Perspectives techniques

Ajout d'autres contraintes (non temporelles)

Comme illustré dans le chapitre 4 de ce mémoire, notre algorithme effectue l'extraction de motifs séquentiels en faisant intervenir une contrainte de support minimum et les trois contraintes temporelles {MinGap, MaxGap, WinSize} qui régissent les distances temporelles entre les items d'une séquence fréquente. Cet algorithme peut être facilement adapté à la prise en compte de contraintes syntaxiques comme la présence ou l'absence de certains items dans les motifs extraits ou à la considération de contraintes de cardinalité sur les itemsets constituant ces motifs. Cependant, la gestion de contraintes exprimées sous forme d'expressions régulières et de contraintes d'agrégats (moyenne, somme, …etc.), qui représentent des contraintes intéressantes du point de vue de la signification des résultats retournés, n'est pas aussi directe. C'est pourquoi, il pourrait être intéressant de faire évoluer notre algorithme afin qu'il puisse intégrer efficacement ce type de contraintes.

Stratégies de parallélisme

Comme le montrent les discussions menées dans le chapitre 3 de ce mémoire, nous avons pu constater que très peu de travaux ont été réalisés pour la parallélisation des algorithmes d'extraction de motifs séquentiels passés en revue. Il sera donc possible d'effectué une étude sur la parallélisation de notre algorithme. Même si pour l'instant cette étude est limitée à l'analyse de la faisabilité de cette tâche, il sera possible de répartir notre arbre BVPT sur différentes machines en se basant sur les préfixes communs que présentent les séquences qu'il représente. Et d'après les études effectuées des premiers résultats ont montré que ceci peut être pratiquement réalisé au moment de la création d'un nouveau nœud (choix des nœuds de meilleurs supports dans les niveaux bas de l'arbre) pour être affecté à une nouvelle tâche qui sera appelée sur ce nœud pour s'exécuter en parallèle. Nous considérons qu'un tel *partitionnement* de l'espace de recherche permettra certainement d'améliorer les performances de notre algorithme, ce qui peut par conséquent constituer un axe de travail très prometteur.

Recherche de motifs séquentiels clos

Malgré les efforts fournis pour améliorer les temps d'extraction, la plupart des algorithmes de recherche de motifs séquentiels présentent une limite inhérente au nombre de motifs. Comme effet de bord, ils peuvent générer un nombre exorbitant de séquences fréquentes. Certains travaux inspirés d'approches proposées pour la recherche de motifs fréquents dans des bases de données sans composante temporelle ont eu pour objectif de réduire l'ensemble des motifs recherchés à un sous ensemble de motifs dits *clos* (*motifs séquentiels fermés*). En pratique, la taille de ces ensembles est beaucoup plus petite, ce qui permet de réduire les temps d'extraction. En plus, ils permettent de générer tous les motifs séquentiels et leurs supports en un temps raisonnable. Il serait donc intéressant de réfléchir à l'adaptation de notre structure de données *BVPT* à la recherche de tels motifs.

Bibliographie :

[01] :http://ifacims2019.com/sites/default/files/call-for-papers.pdf

[02]: Walid Jouini〈 Méthodes et techniques d'extraction des connaissances dans les bases de données 〉 Mémoire de DEA Génie des systèmes industriels. Ecolecentrale de Paris, 2003.

[03]: Jiawei Han, MichelineKamber〈 Data mining : Concepts and Techniques 〉 Morgan Kaufmann Publishers, ISBN 1-55860-901-6, San Francisco, March 2000.

[04]: Layadi Kanza〈Extraction de motifs séquentiels〉Mémoire de fin d'étude pour l'obtention du master UNIVERSITE MOHAMED BOUDIAF-M'SILA,2015/2016.

[05]:https://www.petite-entreprise.net/P-2595-83-G1-principales-taches-du-data-mining.html

[06]:N. Y. Chen, D. Zhu 〈 Data mining for Industrial design and Diagnosis .In proceedings of the 1st International〉 Conference on Multisensor Multisource Information Fusion, Las Vegas, 1998.

[07]: Bentoumi Ismail〈Extraction de motifs séquentiels〉Mémoire de fin d'étude pour l'obtention du MASTER Académique UNIVERSITE MOHAMED BOUDIAF-M'SILA,2016/2017.

[08]:B. Agard, A. Kusiak〈 Exploration des bases de données industrielles à l'aide du Data mining – perspectives 〉 9 ème Atelier Inter établissements de Productique et Pôle de Ressources Informatiques pour la Mécanique (AIP−PRIMECA'05), La Plagne, Avril 2005.

[09]:R. Agrawal, R. Imielinski, A. Swami 〈 Mining association rules between sets of items in large Databases . In the International〉 Conference on Management of Data (SIGMOD'93), pp. 207−216, Washington, May 1993

[10] : Nicolas Pasquier 〈 Data mining : Algorithmes d'extraction et de réduction des règles d'association dans les bases de données. Thèse de doctorat. Université Clermont−ferrand II, Janvier 2000.〉

[11] :https://hal.archives ouvertes.fr/file/index/docid/467753/filename/Extraction_de_bases_pour_les_regles_d_association_a_partir_des_itemsets_fermes_frequents_Pasquier_INFORSID_2000.pdf

[12] : Fabian Mörchen〈 Time Series Knowledge Mining . PhD thesis.Philipps University, Marburg, 2006.〉

[13]: Y. Shahar, M. A. Musen〈 Knowledge based temporal abstraction in clinical domains 〉 Journal of Artificial Intelligence in Medicine, Vol. 8, pp. 267−298, July 1996.

[14]:http://www.lirmm.fr/~poncelet/publications/papers/RNTI09Julien.pdf

[15] :https://tel.archives-ouvertes.fr/tel-00203628/document

[16] : R. Agrawal, R. Srikant〈 Mining Sequential Patterns 〉 . In proceedings of the 11th International Conference on Data Engineering (ICDE'95), pp. 3−14, Taipei, March 1995.

[17]:R. Agrawal, R. Srikant〈 Mining sequential patterns : Generalizations and performance improvements 〉 In the 5th International Conference on Extending Database Technology (EDBT'96) pp. 3−17, Avignon, September 1996.

[18] :Céline Fiot〈 Extraction de séquences fréquentes : Des données numériques aux valeurs manquantes 〉 Thèse de Doctorat, Université Montpellier II, Septembre 2007.

[19]:C. Antunes, A. Oliveira〈 Generalization of pattern-growth methods for sequential pattern mining with Gap constraints〉International Conference on Machine Learning and Data mining (*MLDM'*03)pp. 239−251, Leipzig, July 2003.

[20]:J. Pei, J. Han, B. Mortazavi, H. Pinto, Q. Chen, U. Dayal, M. C. Hsu〈 PREFIX−SPAN : Mining sequential patterns efficiently by prefix projected pattern-growth〉 In the 17th International Conference on Data Engineering (*ICDE'*01), Heidelberg, April 2001.

[21]:M. Y. Lin, S. Y. Lee〈 Fast discovery of sequential patterns by memory indexing 〉 . International Conference on DataWarehousing and Knowledge Discovery (*DAWAK'*02), Aix-en-Provence, September 2002.

[22]: Zhenglu Yang〈 Fast algorithms for sequential pattern mining 〉.Doctoral thesis.Department of Information and Communication Engineering.University of Tokyo, 2008.

[23]: Frank Höppner〈 Knowledge discovery from sequential data 〉.PhD thesis.Technical University of Braunschweig (Germany), 2003.

[24]: Jean Marc Adamo⟨ Data mining for association rules and sequential patterns : Sequential and parallel algorithms⟩Springer Verlag Editions, ISBN 978-0387950488, January 2000

[25] : Hassen M'zali⟨Les règles d'association séquentielles⟩Mémoire en vue de l'obtention de la M.Sc .en intelligence d'affaires, avril 2006.

[26] : Extraction de motifs séquentiels : * *INRIA Sophia Antipolis LIRMM - Université de Montpellier II – ISIM EMA/LGI2P – Nîmes*

[27]:F. Masseglia, F. Cathala, P. Poncelet⟨ The PSP approach for mining sequential patterns ⟩. 2nd European Symposium on Principlesof Data mining and Knowledge Discovery (*PKDD'*98), pp. 176−184, Nantes, September 1998

[28]:Mohammed J. Zaki⟨ SPADE : An efficient algorithm for mining frequent sequences ⟩ Machine Learning Journal, Vol. 42(1−2), pp. 31−60, January 2001

[29]:Mohammed J. Zaki⟨ Sequence mining in categorical domains : Incorporating constraints ⟩ In the 7th International Conference on Information and Knowledge Management (*CIKM'*00)pp. 422−429, Virginia, November 2000.

[30]:M. Garofalakis, R. Rastogi, K. Shim⟨Mining sequential patterns with regular expression constraints ⟩ . In the IEEE Transactionson Knowledge and Data Engineering (*TKDE'*02), Vol. 14(3), pp. 530−552, USA, May 2002

[31] :MAINTENACE INDUSTRIELLE/https://www.iri-lyon.com/nos-domaines/maintenance-industrielle

[32] :https://www.appvizer.fr/magazine/operations/gmao/maintenance-predictive-definition

[33]: KOSALA R., BLOCKEEL H., « Web mining research : A survey », *ACM SIGKDD Explorations*, vol. 2, 2000, p. 1-15.

[34]:CON G., YI L., LIU B.,WANG K., « Discovering Frequent Substructures from Hierarchical Semi-structured Data », *Proceedings of the Second SIAM International Conferenceon Data Mining (SDM-2002)*, Arlington, VA, USA, April 2002.

[35]:LENT B., AGRAWAL R., SRIKANT R., « Discovering Trends in Text Databases »,*Proceedings of the 3rd International Conference on Knowledge Discovery in Databasesand Data Mining (KDD'97)*, Newport Beach, California, August 1997.

[36]:ZAKI M., « Mining Data in Bioinformatics », *in NongS Ye (ed.) Handbook of Data Mining*, Lawrence Earlbaum Associates, 2003, p. 573-596.

Résumé :

Dans le domaine de la fouille de données, l'extraction de motifs séquentiels est devenue, depuis son introduction, une technique majeure avec de nombreuses applications (analyse du comportement des consommateurs, la maintenance industrielle, sécurité, etc.).

Il existe de nombreux algorithmes permettant l'extraction de tels motifs. Dans l'ensemble des travaux relatifs existants, les algorithmes sont de deux principales approches selon leur exploration de l'espace recherche. Une sélection de ces algorithmes est présentée dans ce mémoire tout en complétant chaque description de méthode par une discussion à base de critères que nous avons définis à cet effet. Nous aboutissons à une étude comparative approfondie qui constitue notre contribution analytique dans ce domaine. Quant à l'extraction de séquences de pannes fréquentes, nous proposons une nouvelle solution appelée PSBI ***(Prefix tree for Sequential pattern mining using BItmap representation)*** *s'inscrivant comme une extension d'une méthode existante à la considération de contraintes temporelles sur les motifs extraits tout en réduisant le nombre de parcours de la base de données avec une utilisation raisonnable de l'espace mémoire. Une démarche expérimentale est menée en fin de cette étude pour aboutir à des résultats intéressants et exploitables dans le domaine de la maintenance*

✓ **Mots-clés:** fouille de données, règles d'association, motifs séquentiels