République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Mohamed el-Bachir el-Ibrahimi Bordj Bou Arréridj
Faculté des Mathématiques et L'informatiques

MEMOIRE

Présente en vue de l'obtention du diplôme
Master en informatique

Spécialité : Ingénierie de l'informatique décisionnelle.

THEME :

Amélioration de l'algorithme EMMA pour l'extraction des épisodes fréquents.

Présenté Par :

- Benahcene Lemya
- Belaifa Khawla

Soutenue le : 2021

Devant le jury composé de :

Présidente : Mme Benabid Sonia

Examinatrice : Mme Saidani kaouthar

Encadreur : Mr Moussaoui Boubakeur.

Année Universitaire 2020-2021

Remerciements :

Tout d'abord, nous remercions Dieu de nous avoir donné le pouvoir d'apprendre et de faire ce travail.

Nous remercions notre encadreur M. Moussaoui Boubakeur pour son aide, sa disponibilité et les précieux conseils qu'il nous a donnés.

A cette occasion, nous voudrions remercier tous les professeurs qui ont contribué à notre formation et à la qualité de l'éducation qu'ils nous ont donnée.

Nous n'oublions pas nos parents pour leur soutien et leur patience, nos familles et amis qui nous ont soutenus et encouragés, ainsi que toutes les personnes qui nous ont aidés de près ou de loin.

Merci à tous.

Dédicace :

Je dédie ce mémoire

A mes chers parents ma mère et mon père

Pour leur patience, leur amour, leur soutien et leur encouragement

A mes chères frères Said et Hamza

A mes très chères sœurs Linda et Naima

A mon binôme belaifa khawla

A ma meilleur amie benchala khawla

A mes amis et mes camarades

Sans oublier mon encadreur Mr·Moussaoui Boubakeur·

Benahcene Lemya·

Dédicace :

C'est avec des sentiments de joie et de fierté que j'ai achevé ce travail de patience et de longue haleine, un mémoire qui vient couronner mes cinq années d'étude universitaire.

Je profite cette occasion qui marquera à jamais mon cursus universitaire pour dédier ce mémoire a tous ceux qui m'ont apporté aide et soutenir en particulier

Mes très chers parents Abderrahmane et Noura

A mon chers frère Mohamed

A mes très chers sœurs Chahrazed et Imane

A mes meilleurs amis khadidja,lemya et salma

Sans oublier mon encadreur Mr.Moussaoui Boubakeur.

Belaifa khawla.

Liste des abréviations :

ADN : Acide Désoxyribo Nucléique.

ACP : Analyse en Composante Principale.

ACM : Analyse des Correspondance Multiples.

K-NN : K-Nearest Neighbours.

EFIM : A Fast and memory effecient algorithm for High-utility Itemset Mining.

FHM : Faster High-utility Itemset Mining.

HUI-Miner : High-Utility Itemset- Miner.

HUP-Miner : Hight-Utility Pattern-miner.

ULB-Miner : Utility List Buffer-Miner.

IHUP : Incremental High-Utility Pattern.

FHM + : Faster High-Utility itemset Mining +.

FCHM_bond : Fast Correlated High-utility itemset Miner_bond.

FHN : Faster High-Utility itemset miner with Negative unit profits.

HUINIV-Mine : High Utility itemsets with Negative Item Values- Mine.

HUIM-GA : High-Utility Itemset Mining- Genetic Algorithm.

HUIM-BPSO : High-Utility Itemset Mining- Binary Particle Swarm Optimization.

HUIM-GA-tree : High-Utility Itemset Mining- Genetic Algorithm-tree.

HUIM-BPSO-tree : High-Utility Itemset Mining- Binary Particle Swarm Optimization-tree.

HUIF-PSO : High-Utility Itemset using optimal value Framework- Particle Swarm Optimization.

HUIF-GA : High-Utility Itemset using optimal value Framework- Genetic Algorithm.

HUIF-BA : High-Utility Itemset using optimal value Framework- Bat Algorithm.

HUIM-ABC : High-Utility Itemset Mining based on the Artificial Bee Colony algorithm.

SPP-Growth : Stable Periodic-frequent Patterns- Growth.

PFPM : Periodic Frequent Patterns with Novel Periodicity Measures.

PHM : Periodic High-Utility Itemsets Mining.

MPFPS_BFS : Mining Periodic Frequent Pattern Sequence – Breadth-First search.

MPFPS_DFS : Mining Periodic Frequent Pattern Sequence – Depth-First search.

MRCPPS : Mining Rare Correlated Periodic Patterns Common to multiple Sequences.

SAX : Symbolic Aggregate Approximation**.**

SVM : Support Vector Machine.

Tf * IDF : Term Frequency- Inverse Document Frequency.

TKE : Top-K Episode.

EMMA : Episodes Mining using Memory Anchor.

FIMA : Frequent Itemset mining using Memory Anchor.

POERM : Partially-Ordered Episode Rules Mining.

HUE-SPAN : High Utility Episodes mining by Spanning prefixes.

US-SPAN : Utility episodes in a Sequence of complex events by Spanning prefixes.

TUP : Top-k high Utility ePisode.

EDI : Environnement Développement Intégré.

CDDL : Common Développent and Distribution License.

XML : Extensible Markup Language.

PHP : Hypertext Preprocessor.

HTML : HyperText Markup Language.

IDE : Integrated Development Environment.

JDK : Java Development Kit.

MLF : Motif Local Fréquent.

SDD : Séquence De Données.

LBP : Liste de Borne Projeté.

BDT : Base de Données Temporelle.

EF : Episode Fréquent.

Liste des tableaux :

Liste des figures :

Table des matières

Introduction générale

Des données numériques de plus en plus volumineuses, de plus en plus variées et provenant de plus en plus de sources différentes apparaissent et sont désormais disponibles. 2,5 trillions d'octets de données sont générés chaque jour provenant de nombreuses sources comme le Web, les réseaux de télécommunications, etc. Ces données représentent une mine d'information et c'est la raison pour laquelle il devient primordial de les exploiter. [1]

La fouille de données est le domaine qui consiste à traiter, gérer et créer des modèles de données à partir de ces grandes quantités de données. Le processus traditionnel de fouille de données suit plusieurs étapes : collecter les données, les traiter, les transformer et appliquer des algorithmes dits de fouille dans le but de les analyser et d'identifier des associations (des liens) cachées dans ces données. [1].

Les séquences d'événements sont un type de données fondamental, que l'on trouve dans de nombreux domaines. Par exemple, une séquence peut modéliser des achats quotidiens effectués par un client, un ensemble de lieux visités par un touriste, ensemble de données concerne la consommation d'électricité, ensemble de données liées à la physique, un texte (une séquence de mots), une liste chronologique d'alarmes générées par un système informatique, et des déplacements dans un jeu tel comme aux échecs. Plusieurs tâches d'extraction de données ont été proposées pour trouver des motifs dans des séquences d'événements, tel que l'extraction périodique de motif (Periodic pattern mining), Fouille d'épisode (Episode mining), L'extraction de motifs a haute utilité (high-utility pattern mining), extraction de séries chronologiques (time-series mining), Regroupement et classification (clustering and classification), Mais la tâche qui est sans doute la plus populaire de ce type est l'extraction d'épisodes fréquents (FEM), qui consiste à identifier tous les épisodes (sous-séquences d'événements) qui apparaissent au moins quelques instants dans une séquence d'événements [1], et l'objectif de l'extraction d'épisodes est de trouver des relations entre les événements, dans ce travail nous proposons deux algorithmes d'extraction d'épisode fréquent qui sont EMMA et TKE, pour EMMA compte le support en fonction de la fréquence de la tête et il utilisent des ancres de mémoire pour accélérer la tâche de l'extraction, mais TKE trouve le top-k des épisodes les plus fréquents en fonction de la fréquence de la tête et il applique une recherche dynamique pour réduire l'espace de recherche.

Notre travail consiste dans un premier temps à étudier et à comprendre le domaine de Data Mining et ces techniques, en-suite comprendre le fonctionnement des algorithmes d'extraction

d’épisodes fréquents, dans un deuxième temps, à évaluer et à comparer les performances de ces algorithmes en fonction de différents paramètres et l’amélioration de l’algorithme (EMMA), puis quelques tests de comparaisons pour mesure les performances de notre proposition

La structure de ce mémoire est comme suit :

- Le premier chapitre présente les fondamentaux du data Mining et leurs domaines.

- Dans le deuxième chapitre, nous expliquerons le domaine de fouille d’épisode et parlerons sur leurs algorithmes.
- Le troisième chapitre est consacré à la présentation des outils utilisés pour l’implémentation. Il présente également les tests, les résultats et une discussion sur ces résultats et une comparaison entre les deux algorithmes proposés.

1.1. Introduction

Dans ce chapitre on va discuter le concept de la fouille de donnée qui est connue pour être un sujet très important.

Attendu que, le 17 décembre 2013, il rencontrait des responsables informatiques qui comprenaient Tim Cook d'Apple et Eric Schmidt de Google ainsi que des dirigeants de Twitter, Microsoft, Facebook, Salesforce, Netflix, Etsy, Dropbox, Yahoo, Sherpa Global, Comcast, Il s'agit de discuter le sujet des fouilles de données. Dans une lettre datée du 17 janvier 2014, il a de nouveau soulevé ce sujet en appelant à des réformes du système des fouilles de données. Cela confirme l'attention portée au domaine des fouilles de données (Data Mining) au niveau international et mondial.[1]

Nous parlerons dans ce chapitre du domaine des fouilles de données en général et verrons certaines de ses techniques, tels que :

- ✓ Minage périodique de modéle (Periodic pattern mining).
- ✓ L'extraction de motifs à haute utilité (high-utility pattern mining).
- ✓ Extraction de séries chronologiques (time-series mining).
- ✓ Regroupement et classification (clustering and classification).
- ✓ Fouille d'épisode (Episode mining).

1.2. La fouille de donnée (data mining)

1.2.1. Définition

La fouille de donnée C'est le processus d'analyse des données sous différentes perspectives et de découverte des déséquilibres, des modèles et des corrélations dans les ensembles de données qui sont utiles pour prédire les résultats qui vous aident à prendre une décision judicieuse.

1.2.2. Les étapes de la tache de fouille de données [2]

La tâche de fouille de données peut être divisée en plusieurs étapes :

- **Pré-traitement des données** : c'est l'étape de préparation de données qui consiste au nettoyage de données, à la récupération des données pertinentes et à la transformation des données sous la forme appropriée.

- **Modélisation des données :** elle représente l'étape centrale du processus, qui consiste en l'application de méthodes intelligentes dites de fouille pour extraire les connaissances intéressantes, inattendues ou précieuses.

- **Post-traitement des données :** c'est l'évaluation des connaissances extraites pour l'identification des connaissances les plus intéressantes et leur représentation en utilisant des outils de visualisation.

Il est important de noter que les étapes de pré-traitement et de post-traitement de données sont parfois divisées en plusieurs sous-étapes.

Les étapes qu'on a discuté dans cette section peuvent être résumées par la figure 1-1 qui donne une représentation claire aux étapes et sles opérations de chacune d'elle.

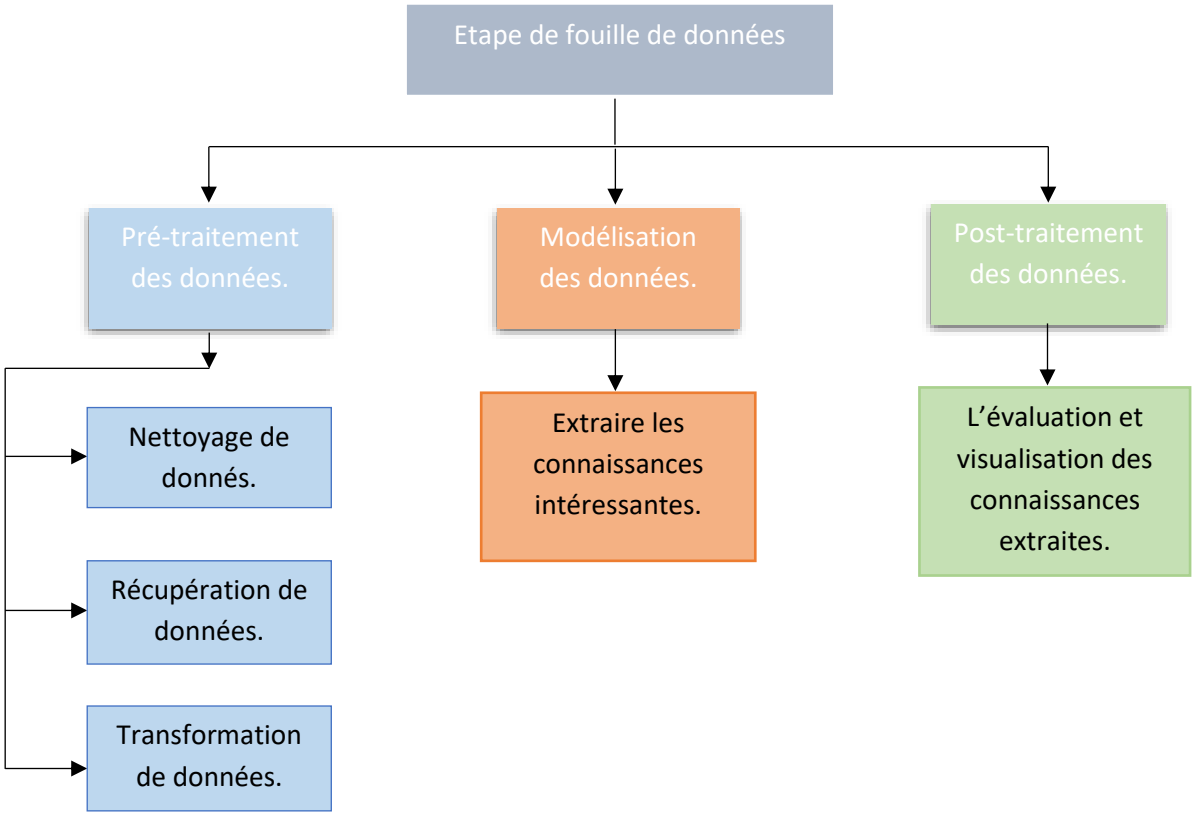


Figure 1-1 : les étapes de la tache de fouille de données.

1.3. Domaines d'application [2]

- **Analyse et gestion des risques, Détection de fraudes Assurance**
  - ✓ Peut-on caractériser les assurés qui font des déclarations d'accident frauduleuses ?
- **Domaine Bancaire cartes de crédit, accord de crédit**
  - ✓ Détecter l'utilisation de cartes de crédit frauduleuse.
  - ✓ Quels sont les clients "à risque" pour l'accord de crédit ?
- **Télécommunications, Systèmes informatiques, Réseaux**
  - ✓ Quels sont les connexions suspectes.
  - ✓ Existe-t-il des groupes d'individus dangereux ?
- **Santé, Médecine**
  - ✓ Etude de l'influence de certaines médications sur l'évolution d'une maladie.
  - ✓ Recherche des médicaments les plus efficaces.
- **Sécurité informatique Détection traditionnelle basée sur des signatures connues Détecter automatiquement de nouvelles intrusions**
  - ✓ Différencier intrusion et mauvaise utilisation.
  - ✓ Classifier les intrusions.
  - ✓ Chercher des modèles prédictifs de mauvaises utilisations.
  - ✓ Construire des profils normaux, des séquences fréquentes.
  - ✓ Identifier des déviations dans les comportements.
- **Biologie -Génomique Analyse des données d'expression de biopuces(microarrays)**
  - ✓ Identifier des similarités dans des séquences d'ADN.
  - ✓ Rechercher le rôle de certains gènes dans une pathologie.
  - ✓ Rechercher le rôle de certains gènes dans l'effet de médications.
  - ✓ Rechercher des gènes qui s'expriment de la même manière.

1.4. Principe de la fouille de donnée (Data Mining) [3]

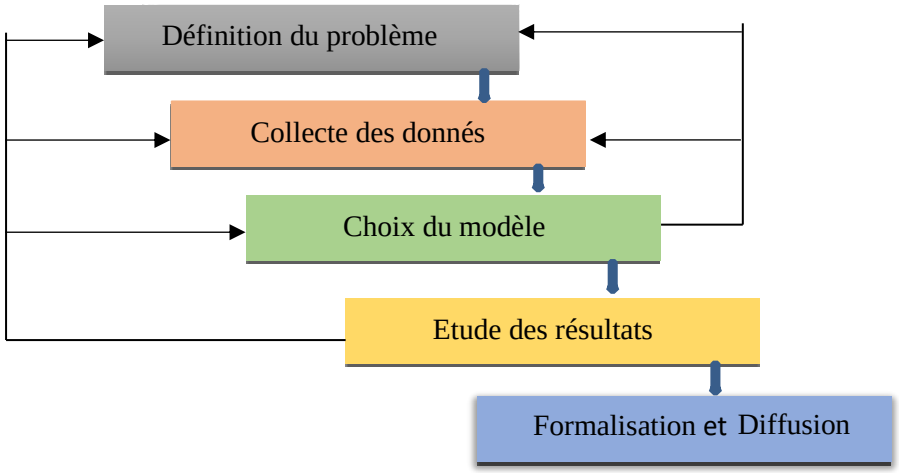


Figure 1-2 : principe du data mining. [3]

- ✓ **Définition du problème :** à ce niveau on doit répondre aux questions du genre : Quel est le but de l'analyse, que recherche-t-on ? Quels sont les objectifs ? Comment traduire le problème en une question pouvant servir de sujet d'enquête pour cet outil d'analyse bien spécifique ? A ce sujet, se souvenir que l'on travaille à partir des données existantes, la question doit être ciblée selon les données disponibles.

- ✓ **Collecte des données :** Une phase absolument essentielle où on n'analyse que des données utilisables, c'est à dire « propres » et consolidées. On n'hésitera pas à extraire de l'analyse les données de qualité douteuse. Bien souvent, les données méritent d'être retravaillées. S'assurer au final que la quantité de données soit suffisante pour éviter de fausser les résultats. Cette phase de collecte nécessite le plus grand soin.
- ✓ **Construire le modèle d'analyse :** Ne pas hésiter à valider le choix d'analyse sur plusieurs jeux d'essais en variant les échantillons. Une première évaluation peut nous conduire à reprendre le point (1 ou 2).
- ✓ **Etude des résultats :** Il est temps d'exploiter les résultats. Pour affiner l'analyse on n'hésitera pas à reprendre le point (1, 2 ou 3) si les résultats s'avéraient insatisfaisants.
- ✓ **Formalisation et diffusion :** Les résultats sont formalisés pour être diffusé. Ils ne seront utiles qu'une fois devenus une connaissance partagée. C'est bien là l'aboutissement de la démarche. C'est aussi là que réside la difficulté d'interprétation et de généralisation.

1.5. Classification des méthodes de la fouille de données [4]

Pour arriver à exploiter les quantités importantes de données, la fouille de donnée utilise des méthodes d'apprentissage automatiques. Ces méthodes sont de deux types : les méthodes descriptives et les méthodes prédictives.

✓ Les méthodes descriptives (recherche de patterns)

Les méthodes descriptives permettent d'organiser, de simplifier et d'aider à comprendre l'information sous-jacente d'un ensemble important de données. Elles permettent de travailler sur un ensemble de données, organisées en instances de variables, dans lequel aucune des variables explicatives des individus n'a d'importance particulière par rapport aux autres. Elles sont utilisées par exemple pour dégager, d'un ensemble d'individus, des groupes homogènes en typologie, pour construire des normes de comportements et donc des déviations par rapport à ces normes telles que la détection de fraudes nouvelles ou inconnues à la carte bancaire ou à l'assurance maladie, pour réaliser de la compression d'informations ou d'images, etc.

Parmi les techniques et algorithmes utilisés dans l'analyse descriptive, on cite :

- Analyse factorielle (ACP et ACM).
- Méthodes des centres mobiles.
- Classification hiérarchique.
- Classification neuronale (réseau de Kohonen).
- Recherche d'association.

✓ Les méthodes prédictives (modélisation)

La raison d'être des méthodes prédictives est d'expliquer ou de prévoir un ou plusieurs phénomènes observables et effectivement mesurés. Concrètement, elles vont s'intéresser à une ou plusieurs variables définies comme étant les cibles de l'analyse. Par exemple, l'évaluation de la probabilité pour qu'un individu achète un produit plutôt qu'un autre, la probabilité pour qu'il réponde à une opération de marketing direct, celles qu'il contracte une malade particulière et en guérisse, les chances qu'un individu ayant visité une page d'un site web y revienne, sont typiquement des objectifs que peuvent atteindre ces méthodes.

En exploration des données prédictives, il y a deux types d'opération : la discrimination ou classement, et la régression ou prédiction, tout dépend du type de variable à expliquer. La discrimination s'intéresse aux variables qualitatives, tandis que la régression s'intéresse aux variables continues. Les méthodes de classement et de prédiction permettent de séparer des individus en plusieurs classes. Si la classe est connue au préalable et que l'opération de classement consiste à analyser les caractéristiques des individus pour les placer dans une classe, la méthode est dite « supervisée ». Dans le cas contraire, on parle de méthode « non-supervisée ». Ce vocabulaire étant issu de l'apprentissage automatique.

La différence entre les méthodes descriptives de classification que l'on a vues précédemment, et les méthodes prédictives de classement provient du fait que leur objectif est divergent : les premières « réduisent, résument, synthétisent les données » pour donner une vision plus claire en vue de la prédiction des valeurs de ces cibles pour les nouveaux arrivants. Parmi les techniques et algorithmes utilisés dans l'analyse prédictive, on cite :

- Arbre de décision
- Réseaux de neurones
- Régression linéaire
- Analyste probabiliste.

Les techniques utilisées :

- K-moyennes.
- Apriori.
- K-NN.
- Réseaux de neurones.
- Algorithmes génétiques.
- Chaine de Markov cachée.
- Arbre de décision.
- Réseaux bayésiens.
- Soft computing: ensembles flous.

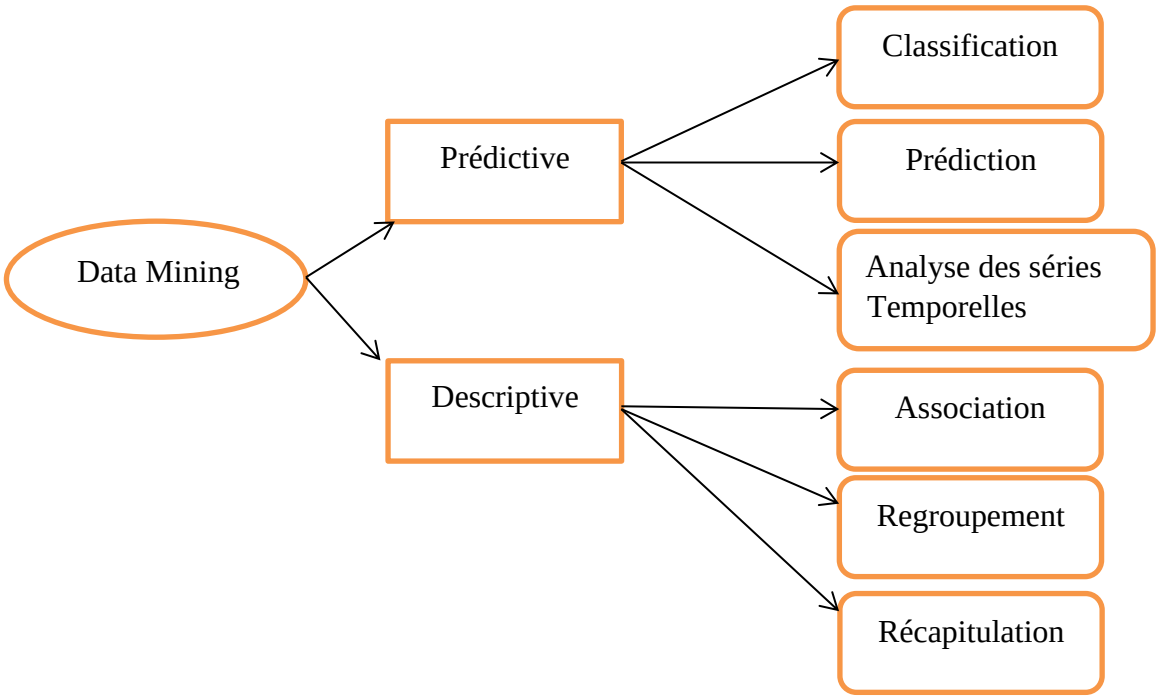


Figure 1-3 : Type des méthodes de Data Mining. [4]

1.6. Les techniques de la fouille de données

Il existe plusieurs techniques dans le domaine de la fouille de données tels comme :

- ✓ Minage périodique de modéle (Periodic pattern mining).
- ✓ L'extraction de motifs à haute utilité (high-utility pattern mining).
- ✓ Extraction de séries chronologiques (time-series mining).
- ✓ Regroupement et classification (clustering and classification).
- ✓ Fouille d'épisode (Episode mining).

1.6.1. L'extraction de modèles à haute utilité

1.6.1.1. Définition

L'extraction de modèles à haute utilité est l'un des problèmes de recherche les plus importants dans l'exploration de données en raison de sa capacité à prendre en compte les valeurs de fréquence non binaires des éléments dans les transactions et les différentes valeurs de profit pour chaque élément. D'autre part, l'exploration de données incrémentielle et interactive offre la possibilité d'utiliser des structures de données et des résultats d'exploration antérieurs afin de réduire les calculs inutiles lorsqu'une base de données est mise à jour ou lorsque le seuil minimum est modifié.[5]

1.6.1.2. Les algorithmes de l'exploration de modèles à haute utilité [8]

- **Algorithmes d'extraction d'ensembles d'éléments à** haute utilité **dans une base de données de transactions** contenant des **informations sur les bénéfices.**
  - ✓ L'algorithme **EFIM** (**Zida et al.2016, Zida et al., 2015**).
  - ✓ L'algorithme **FHM** (**Fournier-Viger et al., 2014**).
  - ✓ L'algorithme **HUI-Miner** (**Liu & Qu, 2012**).
  - ✓ L'algorithme **HUP-Miner** (Krishnamoorthy, 2014).
  - ✓ L'algorithme **mHUIMiner** (Peng et al., 2017).
  - ✓ L'algorithme **UFH** (**Dawar et al, 2017**).
  - ✓ L'algorithme **HMiner** (Krishnamoorty, 2017).
  - ✓ L'algorithme **ULB-Miner** (**Duong et al, 2018**).
  - ✓ L'algorithme **IHUP** (**Ahmed et al., 2009**).
- **Algorithme pour extraire efficacement des ensembles d'éléments à haute utilité avec des contraintes de longueur dans une base de données de transactions.**
  - ✓ L'algorithme **FHM +** (**Fournier-Viger et al, 2016**).
- **Algorithme pour l'extraction d'ensembles d'éléments corrélés à haute utilité dans une base de données de transactions.**
  - ✓ L'algorithme **FCHM_bond**, pour utiliser la mesure de liaison (**FournierViger et al, 2016**).
  - ✓ L'algorithme **FCHM_allconfidence**, pour utiliser la mesure de toute confiance (**Fournier-Viger et al, 2016**).

- **Algorithme d'extraction d'ensembles d'éléments à haute utilité dans une base de données de transactions contenant des valeurs de profit unitaires négatives**.
    - ✓ L'algorithme **FHN** (**Fournier-Viger et al., 2014**).
    - ✓ L'algorithme **HUINIV-Mine** (**Chu et al., 2009**).
- **Algorithme d'extraction d'ensembles d'éléments à haute utilité dans une base de données de transactions à l'aide d'algorithmes évolutifs**
    - ✓ L'algorithme **HUIM-GA** (**Kannimuthu et al., 2014**).
    - ✓ L'algorithme **HUIM-BPSO** (**Lin et al, 2016**).
    - ✓ L'algorithme **arbre HUIM-GA (Lin et al, 2016**).
    - ✓ L'algorithme **HUIM-BPSO-tree** (**Lin et al, 2016**).
    - ✓ L'algorithme **HUIF-PSO** (**Song et al., 2018**).
    - ✓ L'algorithme **HUIF-GA** (S **ong et al., 2018**).
    - ✓ L'algorithme **HUIF-BA** (**Song et al., 2018**).
    - ✓ L'algorithme **HUIM-ABC** (**Song et al., 2018**).

1.6.2. Minage périodique de modèles

1.6.2.1. Définition [7]

Le modèle périodique est une tâche importante d'exploration de données, car des modèles peuvent apparaître périodiquement dans tous les types de données, et il peut être souhaitable de les trouver pour comprendre les données afin de prendre des décisions stratégiques.

Par exemple, les transactions des clients effectuées dans les magasins de détail. L'analyse du comportement des clients peut révéler que certains clients ont des comportements périodiques tels que l'achat de certains produits chaque jour comme le pain et le lait. La découverte de ces modèles peut être utile pour promouvoir des produits le quotidien ou prendre d'autres décisions marketing.

L'analyse du marché boursier est une autre application de modèles périodiques. On peut analyser les fluctuations de prix des stocks pour découvrir les changements périodiques sur le marché. Par exemple, le cours des actions d'une entreprise peut suivre certains modèles chaque mois avant de verser son dividende à ses actionnaires, ou avant la fin d'année.

1.6.2.2. Les algorithmes de Minage périodique de modèles [8]

Ces algorithmes découvrent des modèles qui apparaissent périodiquement dans une séquence d'enregistrement**s** (par exemple des transactions) :

- **Algorithmes pour trouver des modèles périodiques dans une seule séquence d'événements complexes (également appelée base de données de transactions)**
  - ✓ L'algorithme **SPP-Growth** (**Fournier-Viger et al.2019**) pour extraire des modèles périodiques stables dans une base de données de transactions avec ou sans horodatage.
  - ✓ L'algorithme **PFPM** (**Fournier-Viger et al, 2016a**) pour extraire des modèles périodiques fréquents dans une séquence de transactions (une base de données de transactions))
  - ✓ L'algorithme **PHM** (**Fournier-Viger et al, 2016b**) pour extraire des modèles périodiques à haute utilité (modèles périodiques qui génèrent un profit élevé) dans une séquence de transactions (une base de données de transactions) contenant des informations d'utilité.

- **Algorithmes pour trouver des modèles périodiques dans plusieurs séquences d'événements.**
  - ✓ Les algorithmes **MPFPS_BFS** (**Fournier-Viger, P., Li, Z., et al., 2019**) pour extraire des motifs périodiques communs à plusieurs séquences
  - ✓ Les algorithmes **MPFPS_DFS** (**Fournier-Viger, P., Li, Z., et al., 2019**) pour l'extraction de modèles périodiques communs à plusieurs séquences.
  - ✓ L'algorithme **MRCPPS** pour l'extraction de modèles périodiques corrélés rares communs à plusieurs séquences (**Fournier-Viger et al., 2020**).

1.6.3. L'extraction de séries chronologiques (time-series mining)

1.6.3.1. Définition [3]

Les séries temporelles, appelées aussi séries chronologiques, sont des collections de mesures ordonnées dans le temps, et constituent une manière structurée pour représenter des données. Donc une série temporelle est une liste de dates, dont chacune est associée à une valeur. Visuellement, il s'agit d'une courbe qui évolue au fil du temps. Par exemple, les ventes quotidiennes d'un produit peuvent être représentées sous forme de série temporelle.

Ce sont des donnés mesurées à des intervalles de temps régulier. On peut les voir comme des suites d'observations répétées d'un même phénomène à des dates différentes (par exemple la température moyenne journalière en un lieu donné, la consommation moyenne en électricité chaque mois, le prix du baril de pétrole chaque jour...). Les dates sont souvent équidistantes (séries mensuelles, trimestrielles ou annuelles) sauf dans quelques cas (cas de données journalières en économie pas toujours disponibles les jours non ouvrables). On représente habituellement une série temporelle *(xt)* tel que *{1<t<T}* (*t* : le numéro de l'observation) à l'aide d'un graphique avec en abscisse les dates et en ordonnée les valeurs observées

- ✓ Une variable qui représente une ou plusieurs mesures : X = (x_1; x_2;… x_m).

Les valeurs de la variable X à l'instant t sont notéesX_t.

- ✓ Une série temporelle S est une suite ordonnée des valeurs de mesures d'une seule variable X, notée S = < X_{t1}; X_{t2};… X_{tn} >.

On représente généralement une série temporelle par des représentations graphiques où les abscisses indiquent le temps et les ordonnées indiquent les valeurs du phénomène étudié. Si la série est stable autour de sa moyenne donc elle est une série stationnaire, sinon non stationnaire, et quand on aura un phénomène qui se reproduit à des périodes régulières, donc ce phénomène est saisonnier.

1.6.3.2. Domaines d'application [3]

On trouve des exemples de séries temporelles dans de très nombreux domaines.

- ✓ **Finance et économétrie :** évolution des indices boursiers, des prix, des données économiques des entreprises, des ventes et achats de biens, des productions agricoles ou industrielles.
- ✓ **Assurance :** analyse des sinistres.

- ✓ **Médecine et Biologie :** suivie des évolutions des pathologies, analyse d'électroencéphalogrammes et d'électrocardiogrammes, suivie de maladies cancéreuses.
- ✓ **Science de la terre de l'espace :** indices de marées, variations des phénomènes physiques (météorologie), évolution des taches solaires, phénomènes d'avalanches.
- ✓ **Traitement de signal :** signaux de communications, de radars, sonars, analyse de la parole.
- ✓ **Traitement de données :** mesures successives de position ou de direction d'un objet mobile (trajectographie).

1.6.3.3. Algorithmes d'extraction des séries chronologique [8]

Ces algorithmes effectuent diverses tâches pour analyser les données de séries chronologiques

- ✓ Un algorithme pour convertir une série temporelle en une séquence de symboles en utilisant la représentation SAX de séries temporelles. Notez que si l'on convertit un ensemble de séries temporelles avec SAX, il obtiendra une base de données de séquences, qui permet ensuite d'appliquer des algorithmes traditionnels pour l'exploration séquentielle de règles et l'exploration séquentielle de modèles sur des séries temporelles (SAX, 2007).
- ✓ Algorithmes de calcul de la moyenne mobile à précédente d'une série chronologique (pour supprimer le bruit).

Le calcul de la moyenne mobile précédente est un moyen simple mais populaire de lisser une série chronologique pour supprimer le bruit. Il prend comme paramètre une taille de fenêtre w (un nombre de points de données). Ensuite, pour une série chronologique, il remplace chaque point de données par la moyenne de ses w points de données précédentes.

Quelle est l'entrée de cet algorithme (input) ? [9]

L'entrée est une ou plusieurs séries chronologiques. Une série chronologique est une séquence de nombres décimaux à virgule flottante (valeurs doubles). Une série chronologique peut également avoir un nom (une chaîne).

Les séries chronologiques sont utilisées dans de nombreuses applications. Un exemple de série chronologique est le prix d'une action sur le marché boursier au fil du temps. Un autre exemple est une séquence de relevés de température recueillis à l'aide de capteurs.

- **Quelle est la sortie (output) ? [9]**

La sortie est la moyenne mobile de la série temporelle reçue en entrée. La moyenne mobile est calculée en remplaçant chaque point de données dans chaque série chronologique par la moyenne de w points de données précédents dans la même série temporelle.

✓ Un algorithme pour diviser une série chronologique en un nombre donné de segments.

1.6.4. Regroupement et classification (clustering and classification)

1.6.4.1. Regroupement (clustering)

a. Définition

Le clustering (qui rentre dans la catégorie de l'apprentissage non supervisé) est une méthode d'analyse statistique utilisée pour organiser des données brutes en silos homogènes. A l'intérieur de chaque grappe, les données sont regroupées selon une caractéristique commune. L'outil d'ordonnancement est un algorithme qui mesure la proximité entre chaque élément à partir de critères définis.[10]

b. Objectif

Le clustering sert principalement à segmenter ou classifier une base de données (par exemple trier des données clients type âge, profession exercée, lieu de résidence, etc., pour optimiser la gestion de la relation client) ou extraire des connaissances pour tenter de relever des sous-ensembles de données difficiles à identifier à l'œil nu.[10]

c. Applications [4]

- Reconnaissance de formes.
- Analyse des données spatiales.
- Traitement d'image.
- Recherche d'information :

✓ Catégorisation de documents ou de termes

✓ Visualisation de l'information et interfaces de recherche d'information

- Web Mining :

  - ✓ Clustering des usages du web pour découvrir des groupes d'accès similaires
  - ✓ Personnalisation du Web.

d. Méthodologies [4]

Deux méthodologies générales : Algorithmes de partitionnement, Algorithmes hiérarchiques.

- Partitionnement :
  - ✓ Diviser un ensemble de N items en K clusters.
- Hiérarchique :
  - ✓ Par agglomérations : les paires d'items ou de clusters sont successivement liés pour produire des clusters plus grands (bottom-up).
  - ✓ Par divisions : commencer par l'ensemble entier comme cluster et successivement diviser en de plus petites partitions (top-down).

e. Les Algorithmes [8]

Ces algorithmes trouvent automatiquement des clusters dans différents types de données :

- ✓ L'algorithme **K-Means original** (MacQueen, 1967).
- ✓ L'algorithme **Bisecting K-Means** (Steinbach et al, 2000).
- ✓ Algorithmes pour le clustering basé sur la densité :
  - L'algorithme **DBScan** (Ester et al., 1996).
  - L'algorithme Optics pour extraire un ordre de cluster de points, qui peut ensuite être utilisé pour générer des clusters de style DBScan et plus (Ankerst et al, 1999).
- ✓ Un algorithme de clustering hiérarchique.
- ✓ Un outil appelé Cluster Viewer pour visualiser les clusters.
- ✓ Un outil appelé Instance Viewer pour visualiser l'entrée des algorithmes de clustering.

1.6.4.2. La classification

a. Définition [11]

La classification (qui rentre dans la catégorie de l'apprentissage supervisé) est le processus qui consiste à trouver un modèle ou une fonction qui distingue les catégories (appelées étiquettes) de classes présentes dans les données, à partir des attributs qui caractérisent ces données. Prenons un exemple dans le domaine bancaire. Un banquier veut pouvoir prédire si son client sera capable de rembourser son prêt. Pour cela il collecte des informations sur son client : âge, profession, revenus, etc. Puis, en fonction de ces observations, il accepte ou refuse l'octroi du prêt.

La règle d'octroi du prêt est apprise au préalable à partir des clients dont les caractéristiques ainsi que l'étiquette "prêt remboursé" sont connues. En utilisant la fonction de classification, un nouveau client est classé dans une des deux classes : apte à rembourser ou ne pas rembourser le prêt.

Quelle que soit la méthode de classification, un problème important peut influencer la qualité du modèle : le manque de données disponibles pour réaliser l'apprentissage ou la présence de données inadéquates (incertaines ou imprécises), qui empêche la construction d'un modèle fiable.

Il y a plusieurs familles de méthodes de classification. Parmi les familles de méthodes les plus connues nous citons :

- ✓ Les arbres de décision.
- ✓ Les réseaux de neurones.
- ✓ Les machines à support vecteurs (SVM).
- ✓ Les classificateurs de Bayes.

b. Les algorithmes de classification [8]

- ✓ L'algorithme ID3 pour la construction d'arbres de décision (Quinlan, 1986).
- ✓ Un algorithme de classification des documents texte à l'aide d'une approche de classification Naive Bayes (S.Raghu, 2015).
- ✓ Un algorithme de regroupement de textes en utilisant la mesure tf*idf (S. Raghu, 2015).

1.6.5. Fouille d'épisode (Episode Mining) [11]

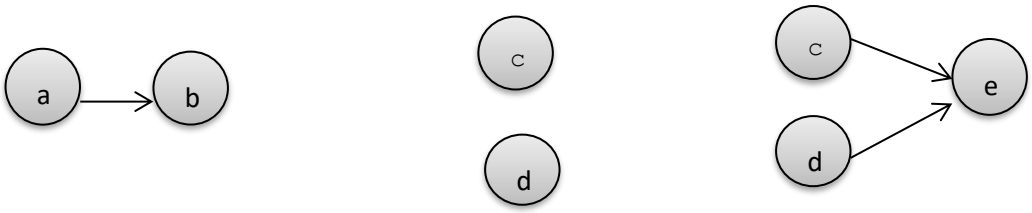
Une dernière technique qu'on traite dans cette section, appelée la fouille d'épisode, elle représente une approche fréquemment utilisée et disponible sur la plateforme open-source ce qui nous ramène à choisir cette technique pour notre projet. On détaille un peu cette approche dans cette section puis on la traite profondément dans le chapitre suivant.

1.6.5.1.Définition

Dans le cas où les données sont formées d'une unique et longue séquence, l'extraction d'épisodes est une tâche essentielle.

Un épisode est un motif temporel composé d'items "relativement proches", qui apparaît souvent tout au long de la séquence ou sur une partie de cette séquence, et est une collection d'événements relativement proches, un épisode peut être comparé à un graphe acyclique où les nœuds représentent les événements et les arêtes représentent la nature de l'ordre entre les événements. Il est important de noter que les motifs peuvent être définis de la même manière que les épisodes, à la différence que dans les épisodes la notion de temps est prise en compte lors de la fouille des épisodes. Selon la nature de l'ordre utilisé, les épisodes peuvent être classés en trois types : les épisodes sériels, les épisodes parallèles et les épisodes composés.

1.6.5.2.Les types d'épisodes



Épisode sériel (1) Épisode parallèle (2) épisode composé. (3)

Figure 1-4 : Les types d'épisode. [11]

Dans la figure 1-4 Un épisode P est une liste partiellement ordonnée d'événements. Il existe trois types d'épisodes.

- **L'épisode (1) est un épisode en série :** il se produit dans une séquence uniquement s'il y a des événements des types a et b qui se produisent dans cet ordre dans la séquence. Dans la séquence, d'autres événements peuvent se produire entre les deux. La séquence d'alarme, par exemple, est fusionnée à partir de plusieurs sources, et il est donc utile que les épisodes soient insensibles aux événements intermédiaires.
    - ✓ P = < p_1 ; p_2 ; … ; p_k > sur I^k avec p ⊆ I. L'ordre entre les événements est total. Le premier événement p_1 de l'épisode représente son préfixe et le dernier événement p_k représente son suffixe.

- **L'épisode (2) est un épisode parallèle :** aucune contrainte sur l'ordre relatif de c et d n'est donnée.
    - ✓ P = {p_1 ; p_2 ; … ; p_k} sur I^k avec p ⊆ I. Il n'y a pas d'ordre entre les événements qui composent l'épisode.

- **L'épisode (3) est un épisode composé :** l'épisode est partiellement sériel et partiellement parallèle, c'est-à-dire qu'il se forme récursivement par une composition sérielle et parallèle des événements.

Les deux types d'épisodes les plus fouillés et utilisés dans la pratique sont les épisodes sériels et les épisodes parallèles.

1.6.5.3.Les algorithmes d'extraction d'épisodes

L'extraction d'épisodes dans des séquences complexes est une problématique récente qui nécessite un algorithme adapté pour prendre en compte l'existence de plusieurs items à chaque temps.

Ces algorithmes découvrent des modèles (épisodes) qui apparaissent dans une seule séquence d'événements.

- Les deux premiers algorithmes proposés dans la littérature pour l'extraction d'épisodes et de règles d'épisode sont *WINEPI* et *MINEPI* qui sont la base de nombreux algorithmes proposés par la suite. Ces deux algorithmes ont trois points communs :
    - ✓ **L'approche pour former les épisodes :** ils commencent tous les deux par l'extraction d'épisodes composés d'un seul événement, et puis étendent itérativement ces épisodes en fusionnant des événements sur leur côté droit. Cette approche est encore utilisée par de nombreux algorithmes récents.

- ✓ **La sélection des épisodes dits fréquents :** les épisodes qui apparaissent suffisamment souvent dans une séquence d'événements, c'est-à-dire les épisodes pour lesquels le nombre d'apparitions dépasse un seuil prédéfini.

- ✓ **L'extraction de régles d'épisode.**

- **Les algorithmes d'extraction d'épisodes fréquents :**

- ✓ L'algorithme **TKE**, qui détecte les épisodes les plus fréquents en fonction de la fréquence de la tête, « c'est le premier algorithme qu'on a étudié, nous parlerons en détail dans le prochain chapitre »
- ✓ L'algorithme **EMMA**, qui compte le support en fonction de la fréquence de la tête, « C'est le deuxième algorithme qu'on a étudié et nous parlerons en détail dans le prochain chapitre »

- ✓ L'algorithme **MINEPI+**, qui compte le support en fonction de la fréquence de la tête.

- ✓ L'algorithme **MINEPI**, qui compte le support basé sur des occurrences minimales, et n'autorise pas les événements simultanés.

- **Les algorithmes pour les règles d'épisode de minage :**

- ✓ L'algorithme traditionnel pour générer des règles d'épisodes standard. Il peut être appliqué pour générer des règles à partir de la sortie de **TKE**, **EMMA** et **MINEPI +**.
- ✓ L'algorithme **POERM** et **POERM-ALL** pour découvrir des règles d'épisode partiellement ordonnées dans une séquence d'événements.

- **Les algorithmes pour l'extraction d'épisodes à haute utilité dans une séquence d'événements complexes (une base de données de transactions) avec des informations sur l'utilité :**

- ✓ L'algorithme **HUE-SPAN** pour l'extraction d'épisodes à haute utilité dans une séquence d'événements complexes (une base de données de transactions) avec des informations sur l'utilité.
- ✓ L'algorithme **US-SPAN** pour l'extraction d'épisodes à haute utilité dans une séquence d'événements complexes (une base de données de transactions) avec des informations sur l'utilité.
- ✓ L'algorithme **TUP** pour extraire les k épisodes les plus utiles dans une séquence d'événements complexes (une base de données de transactions) avec des informations sur l'utilité.

1.7. Conclusion

Dans ce chapitre, nous avons présenté dans un premier temps l'importance du Data Mining, son rôle dans différents domaines et son emplacement dans le processus d'extraction de connaissances à partir des données.

Nous avons présenté ensuite les différentes techniques de la fouille de donnée (Data Mining) et leurs algorithmes.

Il est important de mentionner que les techniques, que nous avons brièvement présentés, ne représentent pas le cœur de notre travail. C'est pour cette raison que nous ne les détaillerons pas.

Parmi ces techniques, nous mettrons en évidence l'extraction d'épisode, dans le prochain chapitre nous détaillerons ce domaine et parlerons de ses différents algorithmes en détail.

2.1. Introduction

Comme nous l'avons mentionnée dans le premier chapitre, il existe de nombreux techniques dans la fouille de donné, notre étude se concentrera sur la technique de la fouille d'épisode, qui à son tour contient de nombreux algorithmes.

La fouille d'épisodes dans des séquences de données complexes est une problématique récente. Elle nécessite un algorithme adapté qui doit prendre en compte l'existence de plusieurs items à chaque instant afin d'extraire les épisodes.

Parmi ces algorithmes, nous avons choisi les quatre suivants (MINEPI, EMMA, TUP, TKE), dont nous parlerons dans ce chapitre.

2.2. Concepts fondamentaux

Afin de comprendre les algorithmes liés à ce domaine et de faciliter leur programmation, nous devons aborder un ensemble de définitions des concepts qu'on a jugé utiles.

2.2.1. Événement [12]

Soit l'ensemble des types d'événements, appelé aussi alphabet, et par souci de simplicité, les types d'événements sont un sous-ensemble fini de $\mathbb{N}$, Donc nous avons : E = {1,2, ..., n}

Un événement α_i est défini comme la paire d'un type d'événement à partir de l'alphabet et de l'heure d'occurrence. L'événement peut contenir plusieurs autres attributs mais dans cet aperçu, nous nous limiterons à cette paire : $\alpha_i = (e_i, t_i)$

Avec $e_i \in E$ et t_i un entier positif indiquant le temps.

2.2.2. Séquence [12]

Une séquence de données peut être présentée sous la forme d'une base de données, d'un fichier journal du système d'alarme, etc. Dans cette formalisation, une séquence est définie comme la succession d'événements triés par leur temps d'occurrence. La séquence elle-même est un triple (s,t_s ,t_e) où t_s et t_e sont respectivement temps de début et temps de fin de la séquence :

$S = \langle \alpha_1\alpha_2\alpha_3 \ldots . \alpha_n \rangle = \langle (e_1, t_1)(e_2, t_2)) (e_3, t_3)\ldots (e_n, t_n)\rangle$

Avec $T_s \leq t_1 \leq t_2 \leq t_3 \leq t_n \leq T_e$

Exemple

(s,30,44)=<(4,31),(5,32),(3,33),(4,33),(3,35),(5,35),(4,37),(5,38),(4,39),(5,39),(3,40),(4,40),(5,40),(3,41),(4,42)〉

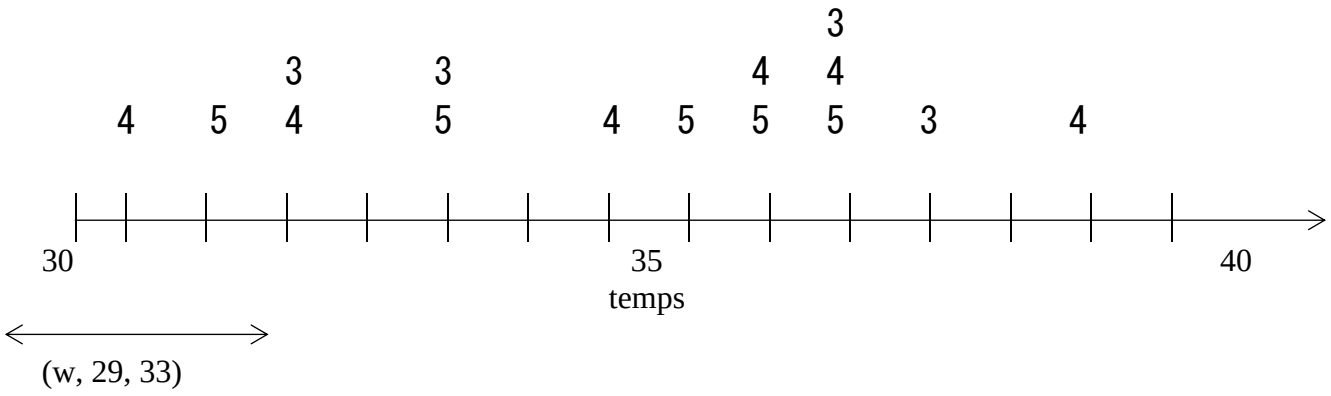


Figure 2-1 : Une représentation d'une séquence (dans cet exemple, certains événements se produisent en même temps).[12]

Nous pouvons voir que les événements se produisent en même temps, il y a des lacunes dans la séquence, (c'est-à-dire des périodes sans événement). Pour pouvoir compter, les occurrences d'événements et les derniers épisodes (qui seront définis plus tard), nous devons définir quand les occurrences comptent et comment les événements sont considérés comme successifs et / ou assez proches. Pour cela, l'utilisateur doit spécifier une fenêtre d'événement et les événements de cette fenêtre peuvent être pris en compte. Nous définissons une fenêtre d'événement comme une séquence d'événements :

W = (w,t_s , t_e) où t_s<t_e et t_e>t_s

Et par cette définition, la fenêtre peut s'étendre en hors de ces séquences comme le montre la figure 2-1.

La fenêtre (w, 29,33), comme le montre la figure 2-1, contient les types d'événements : 4 et 5. Il convient de noter que le les événements (3,33) et (4,33) se sont produits à la fin de la fenêtre mais comme la séquence est définie : ceux-ci ne doivent pas être comptés. Nous appelons la largeur d'une fenêtre sa longueur dans le temps :

(W, 29,34) = ((4,31), (5,32) 〉 , 29,33), largeur (w) = $t_e - t_s$ = 4.

2.2.3. Episode [12]

Auparavant, nous avons défini les séquences et les événements comme la brique primaire de ce formalisme, ils représentent les données brutes, et pour les transformer en connaissance, nous devons mettre en place une partie essentielle de la théorie : l'épisode.

- Un épisode P est une collection ordonnée (ou partiellement ordonnée) de types d'événements se produisant ensemble. Les épisodes peuvent être décrits comme des graphiques acycliques dirigés. Ils peuvent être considérés comme une relation mappant différents types d'événements, créant une relation antécédente / précédent entre chaque événement ou une relation de simultanéité. Et il y a trois types d'épisodes (Nous l'avons mentionné et expliqué dans le premier chapitre 1.5.5.b).

2.2.4. La taille de l'épisode [11]

Soit l'épisode P =< p_1; p_2;…; p_k >. Sa taille représente le nombre de ses événements, elle est notée $|P|$: $|P|$= k.

Soit Q =< q_1; q_2;…; q_s > un autre épisode. La concaténation de l'épisode P et de l'épisode Q forme l'épisode noté P · Q = <p_1; p_2;…; p_k; q_1; q_2;…; q_s > de taille k + s.

2.2.5. Sous-épisode [13]

P est un sous-épisode de Q, noté P ⊆ Q, si $\forall_i\, p_i \subseteq q_i$ (p_i peut être∅). Dans ce cas, Q est appelé un super-épisode de P.

Par exemple, l'épisode P= (< A >, < BD >) est un sous-épisode de Q = (<AF >, < BD >). Au contraire, l'épisode R = (< AB >) n'est pas un sous-épisode de Q.

Episode sans répétition : dans l'épisode (< A >, < AF >) l'item A apparaît plusieurs fois, nous appelons "épisode sans répétition" un épisode qui ne contient pas d'items apparaissant plusieurs fois.

Exemple :

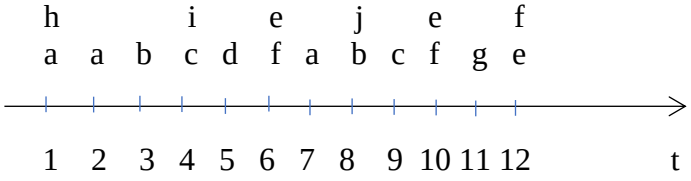


Figure 2-2 : Exemple d'une séquence d'évènement s. [**11**].

L'événement *ef* composé de l'itemset $\{e, f\}$, apparaît trois fois : aux instants : t_6, t_{10} et t_{12}. On peut observer que, dans la séquence d'événements *S*, le motif "l'événement *a* est apparu régulièrement avant l'événement *ef*, aux plus cinq instants avant *ef*", apparaît plusieurs fois. Ce type de motif est un épisode, il peut être noté (< *a ; ef* > ; 5).

- Dans la figure ci-dessus L'épisode *P* =< *ef* > est un sous-épisode de l'épisode

- *Q* =< *a ; ef* >. En revanche, l'épisode *P0* =< *ae* > n'est pas un sous-épisode de *Q*.

2.2.6. L'occurrence d'un épisode [11]

Notée occ_i(P), est la suite des instants (temps) d'apparition dans la séquence des événements qui composent cet épisode. L'occurrence d'un épisode est généralement notée occ_i(P) = (t_{s_i}; t_{e_i}) : les instants d'apparition de son préfixe et de son suffixe, car ce sont les événements les plus importants et le plus souvent utilisés par les algorithmes. La liste de toutes les occurrences d'un épisode P est notée Occ(P).

2.2.7. Le support de l'épisode [11]

Noté supp(P), représente le nombre de ses occurrences qui sont obtenues selon la mesure de fréquence appliquée : supp(P) =|Occ(P) |. L'épisode P est considéré comme un épisode fréquent dans la séquence, Si supp(P) ≥ minsupp, où minsupp est un seuil de support prédéfini.

2.2.8. Règle d'épisode [11]

Une règle d'épisode R est une implication R : P → Q qui signifie que l'épisode Q représente la conséquence de la règle, apparaît régulièrement après l'épisode P qui représente l'antécédent. Le terme "régulièrement" signifie que la conséquence apparaît après l'antécédent avec une probabilité relativement élevée.

Le support de la règle R représente le support de la concaténation des deux épisodes P et Q qui la composent : supp(R) = supp(P·Q).

2.2.9. La confiance de la règle d'épisode

La confiance de la règle d'épisode R : P → Q représente la probabilité que sa conséquence Q apparaisse, sachant que son antécédent P est apparu :

$$\text{Conf (P}\rightarrow Q) = \frac{supp(P.Q)}{supp(P)}$$

Lorsque la confiance de la règle d'épisode dépasse un seuil prédéfini minconf, la règle est considérée comme une règle confiante.

La tâche de fouille de règles d'épisode consiste en règle générale à extraire les règles valides : les règles d'épisode fréquentes et confiantes.

2.3. Algorithmes

Les premiers algorithmes d'extraction d'épisode proposé dans la littérature sont Winepi et Minpi qui seront la base de nombreux autre algorithmes proposés par la suite qui nous avons cité dans le chapitre précédent et maintenant nous avons détailler les quatre suivants : MINEPI, TUP, EMMA et TKE.

2.3.1. L'algorithme MINEPI [14]

MINEPI (Mannila et al., 1997) est le premier algorithme pour extraire des épisodes fréquents dans une séquence d'événements. Un épisode fréquent est une sous-séquence qui apparaît fréquemment dans une certaine fenêtre temporelle dans une longue séquence d'événements.

Différents algorithmes ont été proposés pour trouver des épisodes fréquents dans les données. Il est à noter que différents algorithmes ne comptabilisent pas le support (fréquence d'occurrence) d'un épisode de la même manière. L'algorithme MINEPI compte le support en utilisant le concept d'occurrences minimales.

- L'algorithme MINEPI procède à une première phase de génération d'épisodes candidats puis à une phase d'évaluation du support de ces épisodes.

Définition [11] : l'occurrence minimale : Soit occ(P) = $(t_s; t_e)$ une occurrence de l'épisode P. Cette occurrence est considérée comme une occurrence minimale, notée mo(P), si elle ne contient pas une autre occurrence du même épisode :

Si $\nexists$occ'(P) = $(t'_s; (t'_e)$: $\{(t'_s; t'_e) \subseteq (t_s; t_e)$; $(t_s < t'_s \cap t'_e \leq t_e) \_ (t_s \leq t'_s \cap t'_e < t_e)\}$.

La liste des occurrences minimales d'un épisode P est notée Mo(P), et le support de l'épisode : supp(P) =|Mo(P) |.

2.3.1.1. Les entrées de l'algorithme MINEPI [14]

MINEPI prend en entrée une séquence d'événements avec un support minimum, une fenêtre maximum et un paramètre booléen self_increment, qui doit être défini sur vrai (true) si l'ensemble de données d'entrée n'a pas d'horodatage ou sinon faux (false). Considérons la séquence suivante composée de 10 points temporels (t1, t2…, t10) et de 4 événements de type (1, 2, 3, 4). Cette base de données est fournie dans le fichier texte "contextMINEPI.txt" du package ca.pfv.spmf.tests de la distribution SPMF.

Chaque ligne de la séquence est appelée un ensemble d'événements et se compose de :

1) Un point dans le temps (première colonne).
2) Un événement (deuxième colonne).

Temps	Evénement
t1	1
t2	2
t3	1
t4	2
t5	4
t6	3
t7	2
t8	3
t9	2
t10	1

Tableau 2-1 : Base de données de séquence 01 [14].

2.3.1.2. La sortie de l'algorithme MINEPI

La sortie de MINEPI est l'ensemble des épisodes fréquents ayant un support d'au moins un seuil minSup (un entier positif) défini par l'utilisateur. Pour expliquer ce qu'est un épisode fréquent, il est nécessaire de revoir certaines définitions.

Le support d'un épisode est le numéro de son occurrence minimale.

L'occurrence minimale : Soit occ(P) = (t_s; t_e) une occurrence de l'épisode P. Cette occurrence est considérée comme une occurrence minimale, notée mo(P), si elle ne contient pas une autre sous-intervalle de temps qui contient cet épisode, et cet intervalle de temps n'est pas supérieur au paramètre de fenêtre maximum épisode :

Si ∄occ'(P) = (t'_s; (t'_e) : {(t'_s; t'_e) ⊆ (t_s; t_e) ; ($t_s < t'_s \cap t'_e \leq t_e$) _ ($t_s \leq t'_s \cap t'_e < t_e$)}.

La liste des occurrences minimales d'un épisode P est notée Mo(P), et le support de l'épisode : supp(P) = |Mo(P)|.[11]

- Par exemple, considérons l'épisode < (1), (2)> qui signifie que l'événement 1 a été suivi de l'événement 2. Les occurrences minimales de < (1), (2)> sont [t1, t2] et [t3, t4]. On peut affirmer que [t1, t3] contient également < (1), (2)>, mais [t1, t3] n'est pas une occurrence minimale de < (1), (2)>, car il existe un sous- intervalle de temps [t1, t2] contenant < (1), (2)>. Une occurrence minimale doit satisfaire le seuil de fenêtre maximal fourni par l'utilisateur. [14]

Un épisode fréquent est un épisode dont le support n'est pas inférieur à minSup. Par exemple, si vous définissez minSup = 2 et maxWindow = 2. Nous obtenons 6 épisodes fréquents

2.3.1.3. Extraction des épisodes fréquents [11]

Pour chercher l'occurrence minimale d'un épisode de taille *m* + 1, les occurrences de deux épisodes de taille *m* sont exploitées en effectuant une opération de jointure temporelle.

Soient deux motifs séquentiels *P*, *Q*. La jointure temporelle de ces deux motifs est une opération qui consiste, à partir des listes d'occurrences de ces deux motifs, à créer des occurrences d'un motif séquentiel *P. Q* (la concaténation de *P* et *Q*).

Alors il faut connaitre la définition de la jointure temporelle en général, telle qu'elle doit être définie dans le cas de calcul d'une occurrence minimale.

Une jointure temporelle des listes d'occurrences des deux épisodes P et Q avec :

occ(P) =(t_s; t_e) ∈ Occ(P), occ(Q) = (t'_s; t'_e) ∈ Occ(Q), *est effectuée* comme suit : pour Occ(P); Occ(Q) : Occ(P · Q) = Occ(P) · Occ(Q) = {(t_s; t'_e) : ts < t'_e}.

Pour construire une liste des occurrences minimales d'un épisode P · Q en utilisant une jointure temporelle, et à partir de là, nous nous tournerons vers la définition suivante :

La jointure temporelle avec occurrence minimale est :

Mo (P · Q) =Mo(P) · Mo(Q) = {mo (P · Q) = (t_s; t'_e) : ($t_s < t'_e$) ⋀ (∄mo'(P · Q) : mo'(P · Q) ⊆ mo(P · Q))}.

Dans la séquence d'événements S, considérons les deux épisodes < a > et < e >. Occ(<a >) = {(1; 1); (2; 2); (7; 7)} et Occ(< e >) = {(6; 6); (10; 10); (12; 12)}. Afin de construire la liste d'occurrences de l'épisode < a > · < e >, l'opération de jointure temporelle est effectuée sur les listes d'occurrences de < a > et de < e >, On obtient la liste d'occurrences suivante : Occ(< a > · < e >) = Occ(< a >) · Occ(< e >) = {(1; 6); (1; 10); (1; 12); (2; 6); (2; 10); (2; 12); (7; 10); (7; 12)}. Lorsqu'une occurrence minimale est exigée, la liste des occurrences est réduite et devient : Mo (< a > · < e >) = mo(<a>) · mo (< e >) = {(2 ; 6) ; (7 ; 10)}. (Dans cet exemple occ(< a >) = mo(< a >) et occ(< e >) = mo(< e >), puisque < a > et < e > sont des épisodes composés d'un unique événement).

Algorithme 1 : *MINEPI* : sous-algorithme1 extraction des épisodes fréquents :

Données : S : la séquence d'événements, minsupp : le seuil de support minimal.

Résultat : E : l'ensemble des épisodes fréquents

```
1  Début
2      E = ∅ ;        // initialisation de l'ensemble des épisodes fréquents
3      m = 1 ;        // m représente l'indice de la taille des épisodes
4      Cm = ∅ ;       // initialisation de l'ensemble des candidats de taille 1
5      Calculer C_m = {P : |P| = 1} ;        // génération des candidats de taille 1
6      Tant que ( C_m ≠ ∅) faire        // évaluation du support |Mo(P)| ≥ minsupp
7          Calculer E_m : {P : P ∈ C_m ∧ supp(P) ≥ minsupp}
8          E = E ∩ E_m                  // génération des candidats de taille m + 1
9          Calculer C_m +1 = {P : |P| = m + 1 ∧ ( ∀ P'; P'' ⊂ P ∧ P'; P'' ∈ E_m:
10         Suffixe(P') =préfixe(P''))}
11         Calculer Mo(P) : P ∈ C_m+1
12         m = m + 1
13     Fin tq ;
14 Fin.
```

Une fois les épisodes fréquents identifiés, ces épisodes sont utilisés afin de former les règles d'épisode (fréquentes et confiantes).

2.3.1.4. Extraction des règles d'épisode [11]

Tout comme les règles d'association, l'extraction de règles d'épisode est une tâche simple et se fait traditionnellement en deux étapes :

- La génération des règles d'épisode candidate : formées à partir de l'ensemble des épisodes fréquents (ligne 3). Soit *P* un épisode fréquent avec *P = P'·P'*. Une règle candidate est donc formée de manière à ce que *P' représente* son antécédent et *P''* représente sa conséquence (lignes 5 et 6) : *R* : *P' → P''*. Rappelons que si un épisode *P* est fréquent, alors tous ses sous-épisodes *P'* et *P''* sont fréquents (par la propriété d'anti-mono-tonicité), et que la construction de règles d'épisode à partir des épisodes fréquents assure donc que les règles candidates formées sont fréquentes.[11]

- L'évaluation de la confiance de chaque règle candidate (ligne 7) : si la confiance d'une règle candidate est supérieure à un seuil prédéfini *minconf,* alors elle sera une règle résultat de l'algorithme.

Algorithme 2 : *MINEPI* : sous-algorithme 2 extraction de règles d'épisode

Données : E : L'ensemble des épisodes fréquents, minconf : le seuil de support minimal.

Résultat : $E_{règles}$: L'ensemble des règles d'épisode

```
1  Début
2      E_règles : = ∅   // initialisation de l'ensemble de règles d'épisode
3      Pour chaque (P ∈ E) faire
4          Pour chaque (P' ⊂ P) ;   // génération de candidats faire
5              Calculer P'' : P = P'. P''
6              Si conf (P' → P'') ≥ mincof   // évaluation de candidats alors
7                  E_règles : = E_règles ∩ {P' → P''}
8              Fin si ;
9          Fin pour ;
10     Fin pour ;
11 Fin.
```

2.3.2. L'algorithme TUP

TUP est un algorithme permettant de découvrir les k épisodes les plus utiles dans une séquence d'événements complexes contenant des informations d'utilité (ce qui équivaut à une base de données de transactions triée par le temps).

L'extraction d'épisodes de grande utilité peut être utilisée pour découvrir des sous-séquences d'événements (épisodes) qui ont une grande importance (par exemple, rapportent un profit élevé) dans une séquence d'événements complexe. Une séquence d'événements complexe est une liste ordonnée d'ensembles d'événements, où un ensemble d'événements est un ensemble d'événements se produisant simultanément. Dans une séquence d'événements complexe avec des informations sur l'utilité, chaque événement est annoté avec des informations sur l'utilité (des nombres indiquant l'importance de l'événement - par exemple, le rendement du profit par l'événement).

Deux versions des algorithmes TUP sont proposées dans SPMF, appelées respectivement TUP_Combined et TUP_Preinsertion. Ce sont deux versions utilisant des optimisations différentes, qui produisent le même résultat mais peuvent être plus ou moins efficaces sur des séquences différentes.

2.3.2.1. L'entrée de l'algorithme TUP [15]

TUP prend en entrée une séquence d'événements complexes (également appelée base de données de transactions) avec des informations d'utilité, un paramètre k représentant le nombre de motifs à sortir et une durée minimale (un entier). Considérons la séquence suivante composée de 6 points temporels (t_1, t_2... t_6) et de 7 types d'événements (1, 2, 3, 4, 5, 6, 7). Cette base de données est fournie dans le fichier texte " ExampleTUP.txt " dans le package ca.pfv. spmf.tests de la distribution SPMF.

Points de temps	Événements	Utilitaire d'événement total	Utilitaires de chaque événement
t1	5 3	14	10 4
t2	3	Dix	Dix
t3	5 4	13	10 3
t4	2	4	4
t5	4 1	3	1 2
t6	7 6	5	2 3

Tableau 2-2 : Base de données de séquence 02. [6]

Chaque ligne de la séquence est appelée un ensemble d'événements et se compose de :

- Un point dans le temps (première colonne).
- Un ensemble d'événements qui sont considérés comme simultanés à ce moment (la deuxième colonne du tableau).
- La somme des utilités (par exemple le profit) des événements (la troisième colonne du tableau).
- L'utilité (importance) de chaque événement à ce moment (la quatrième colonne du tableau).
- Notez que la valeur de la troisième colonne de chaque ligne est la somme des valeurs de la quatrième colonne.

Il existe plusieurs applications dans la vraie vie. Une application est une séquence de transactions client. Imaginez que chaque point temporel est un ensemble d'articles achetés par un client à un moment donné. Dans ce cas, un point temporel est une transaction et les événements sont des articles achetés par le client. Ensuite, les utilités des événements sont le profit généré par la vente de chaque article (événement) dans chaque transaction (point temporel). Dans cet exemple, la première transaction (point temporel) nommée " $\boldsymbol{t_1}$ " représente qu'un client a acheté les articles 5 et 3. Le montant d'argent (utilité) dépensé pour chaque article est respectivement de 10 $ et 4 $. Le montant total dépensé dans cette transaction est de 10 + 4 = 14 $.

2.3.2.2. TUP-Basique [16]

Dans cette sous-section, nous présentons un algorithme de base pour découvrir les meilleurs épisodes à haute utilité à partir d'une séquence d'événements complexes (voir l'algorithme 5). L'algorithme prend en entrée une séquence d'événements complexes et un paramètre k. L'algorithme maintient une liste triée de taille K de manière dynamique, qui contient les k épisodes les plus utiles. Le seuil d'utilité minimale (min utilité) stocke l'utilité du k-ème épisode actuel. Le seuil minimal d'utilité est initialisé à zéro avant le début du processus d'extraction car le tampon top-k est vide.

Tout d'abord, la base de données est analysée une fois pour trouver tous les épisodes d'une longueur (ligne2). L'occurrence minimale, l'utilité et l'EWU des épisodes de 1 longueur sont calculés. Si l'EWU d'un épisode est supérieur au seuil minimum (ligne 5), l'algorithme explore l'espace de recherche des épisodes de grande utilité avec l'épisode actuel comme préfixe. L'algorithme suit une stratégie de recherche en profondeur pour trouver de nouveaux épisodes. Pour explorer l'espace de recherche, un épisode est concaténé simultanément (ligne 6) et en série (ligne 7) aux événements. Soit l'occurrence d'un épisode α [Ts, Te]. Les événements se produisant en anneau à Te sont concaténés simultanément à l'épisode α. Alors que les événements survenant entre Te + 1 et Ts + MT D −1 sont enchaînés en série à l'épisode α.

Algorithme 3: TUP-Basic (CES, MT D, k)[16]

Données : Une séquence d'événements complexe CES, nombre d'épisodes souhaité

Résultat : L'ensemble complet des épisodes de grande utilité top-k

```
1  Debut
2      Lire CES pour trouver tout l'épisode d'une longueur (one Length EpiSet) ;
3      Min utilité =0 ;
4      Pour épisode epi dans oneLengthEpiSet faire
5          Si EWU (epi) ≥ min utilité alors
6              Concaténation simultanée (epi, minOcc (epi), min utilité, k) ;
7              Concaténation série (epi, minOcc (epi min utilité, k) ;
8          Fin si ;
9      Fin pour ;
10 Fin.
```

Chaque nouvel épisode, β, formé par la concaténation, appelle à insérer l'épisode (.) Qui met à jour la liste des épisodes Top-K (voir Algorithme 4). L'épisode d'entrée est ajouté à la liste si la taille de la liste est inférieure à celle définie par l'utilisateur ou son EW u n'est pas inférieure au seuil d'utilité minimum. Une fois que K candidats sont découverts, l'utilité minimum est soulevée à la k the utilitaire le plus élevé de la liste I.e. à la moindre utilité de la liste. En outre, seuls les épisodes satisfaisants utilitaires minimum sont insérés dans la liste et le (k + 1) -ème épisode est retiré du tampon. AF TER L'ALGORITHM TERMINATES, LA LISTE TOP-K contient la sortie souhaitée de l'extraction de l'épisode d'utilitaire haut de gamme Haut-K élevé dans la séquence d'événement complexe.

L'algorithme renvoie le résultat correct comme si l'EWU d'un épisode est inférieur à l'utilité de la KTH actuelle épisode, il est garanti qu'aucun super-ensemble de cet épisode Peut être dans le tampon op-K en raison de la fermeture de la fermeture à la baisse. Les épisodes top-k pour k = 3 et mtd = 2 sont : {h (ed), (b) i : 29, H (e), (b) I : 26, H (CE), (c) I : 24} …

Algorithm 4: Insert Episode (*epi; min utility; k*)[16]

Données : Un épisode epi, utilité minimale min utilitaire, souhaitée nombre d'épisodes k

Résultat : Top-K List : Top-k épisodes de grande utilité parmi les candidats

```
Début
    Si taille (T op - KList) <k alors
        Ajouter epi à la liste Top-k.
    Sinon
        Si utilité (epi)> minutiliti alors
            Supprimer le kème épisode de grande utilité, c'est-à-dire l'épisode ayant
            Le moins d'utilité ;
            Ajoutez epi à la liste ;
            Trier la liste par ordre décroissant de valeurs d'utilité d'épisodes ;
            minutiliti = moindre utilité dans la liste Top-K ;
        Fin si ;
    Fin si ;
Fin.
```

2.3.3. L'algorithme EMMA

2.3.3.1. Définition

EMMA (Episodes Mining using Memory Anchor) qui utilisent des ancres de mémoire pour accélérer la tâche d'extraction.

EMMA est un algorithme d'extraction d'épisodes fréquents dans une séquence complexe. Il a la caractéristique d'extraire un ensemble complet d'épisodes. EMMA extrait d'abord les motifs fréquents, pour chaque temps t, dans l'ensemble des items apparaissant à t. Cette première caractéristique permet de diminuer la complexité de l'algorithme en diminuant la grande taille des motifs extraits. Les épisodes sont ensuite construits incrémentalement en prenant en compte la liste de bornes de chaque1-uplet épisode.

2.3.3.2. Les phases de l'algorithme EMMA [13]

L'algorithme EMMA comporte trois phases.

- **Phase 01 : Extraction de motifs fréquents**

EMMA utilise un algorithme appelé FIMA (Frequent Itemset mining using Memory Anchor) pour extraire un ensemble de motifs fréquents locaux : LFP (Local Frequent Patterns). Tous les items dans un motif fréquent ayant le même temps d'apparition t, le motif est dit "local". FIMA commence par une lecture de la séquence de données pour trouver les motifs locaux de taille1. Ensuite, pour chaque motif local, une combinaison est faite avec un autre motif du même temps t. Cette procédure de combinaison est appliquée récursivement afin d'avoir des motifs plus longs. À chaque itération, les motifs sont choisis en fonction de leur support. Ensuite, une liste de bornes de chaque motif local fréquent est créée. Notons qu'un motif fréquent extrait représente un1-uplet épisode.

L'algorithme 5 : FIMA [3].

Procédure FIMA (Base de données temporelle TDB, minsup)

```
1   Début
2       Lire TDB, trouver 1-item fréquent F1 ;
3       Lire TDB, transformer TDB en base de données indexée Index DB et maintenir
4       L'emplacement de tous les f1 dans la base de données d'index ;
5       Pour chaque fi en F1 faire
6           FIM A(fi, fi.listemplacement) ;
7           Sous-procédure FIMA (List des motifs locaux)
8           LFP = motifs fréquents locaux dans motif.PList ;
9           Pour chaque fi dans LFP faire
10              FIMA (motif [fi, fi. Listemlacement]) ;
11          Fin pour ;
12      Fin pour ;
13  Fin.
```

- **Phase 02 : Encodage de la séquence de données**

Une fois l'ensemble des1-uplet épisodes extrait, un identifiant id est associé à chacun (voir tableau 03). La séquence de données (figure 08) est encodée par les id de1-uplet épisodes dans chaque intervalle selon leur liste de bornes (voir tableau 04). L'ensemble des id est appelé ID.

Exemple :

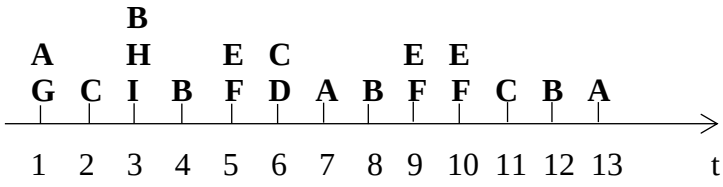


Figure 2-3 : Séquence complexe [13]

ID	l-uplet épisode	Liste de bornes
#1	A	[1,1], [7,7], [13,13]
#2	B	[3,3], [4,4], [8,8], [12,12]
#3	C	[2,2], [6,6], [11,11]
#4	E	[5,5], [9,9], [10,10]
#5	F	[5,5], [9,9], [10,10]
#6	EF	[5,5], [9,9], [10,10]

Tableau 2-3 : encodage des 1-uplet épisodes [13]

Temps	1	2	3	4	5	6	7	8	9	10	11	12	13
Id	#1						#1						#1
			#2	#2				#2				#2	
		#3				#3					#3		
					#4				#4	#4			
					#5				#5	#5			
					#6				#6	#6			

Tableau 2-4 : encodage de la séquence de donnée. [13].

- **Phase 03 : Extraction des épisodes fréquents**

La dernière étape de l'algorithme EMMA extrait les épisodes fréquents en utilisant la séquence de données encodée. Chaque id représente un préfixe d'un épisode à construire (le premier élément de l'épisode). Pour chaque id et selon sa liste de bornes, on cherche les autres id fréquents dans sa liste de bornes projetées définie comme suit :

Liste de bornes projetées : Pour une borne d'un épisode P, une borne projetée représente l'intervalle de temps (de taille au plus w) où les épisodes candidats (super-épisodes de P) sont recherchés. Pour l'épisode P, qui a sa $i^{ème}$borne Borne_i(P) = [t_{s_i}, t_{e_i}], la borne projetée de $Borne_i$ (P) est ProjBorne_i (P) = [t_{s_i}+ 1, t_{s_i}+w−1]. Par exemple, pour w= 6, le 1-uplet épisode #1 a la liste de bornes projetées suivante :

ProjBorne (#1) = {[2,6], [8,12]}. Un nouvel épisode candidat est alors construit en étendant chaque id avec chacun des id fréquents dans sa liste de bornes projetées. Dans ce cas, #1, #2, #3, #4, #5, #6 qui sont fréquents sont tous étendus, pour minsupp = 15% (tableau 2). Pour chaque épisode candidat, sa liste de bornes est ensuite calculée. Si cet épisode est fréquent, il est de nouveau étendu avec un nouvel id. Par exemple, pour construire l'épisode <#1, #2>, sa liste de bornes est Borne (<#1, #2>) = [1,3], [1,4], [7,8], [7,12]. Support (<#1, #2>) = 2/13 = 15.3%, donc<#1, #2> est un épisode fréquent et il sera étendu avec d'autres id. EMMA utilise un parcours en profondeur pour étendre les épisodes, et l'itération s'arrête quand l'épisode candidat n'est pas fréquent ou s'il n'y a plus d'id avec lequel l'étendre. Cette façon de construire les épisodes peut mener à des épisodes contenant plusieurs occurrences du même item.

L'algorithme 6 : EMMA [5]

Procédure EMMA (Base de données temporelle TDB, minsup, maxwin)

```
1   Début
2       Appeler procédure FIMA pour trouver tous les itemsets fréquent F1 et leur
3       Liste de tepmps ;
4       Associer chaque itemset a un #ID et construise la base de données encodée EDB ;
5       Pour chaque fi di dans les ID fréquents F1 faire
6           emmajoin(fi di , fi di.listbornée) ;
7           Procédure emmajoin (épisode, listbornée)
8           Trouver les ID fréquents locaux LFP dans épisode.PBL et leur liste de borne ;
9           Pour chaque fi dans LFP faire
10              Ecrire épisode.fi;
11              Si (EXTCount(fi. listbornée)≥ minsup * |TDB|) alors
12                  emmajoin (épisode. fi, fi. listbornée) ;
13              Fin si ;
14          Fin pour ;
15      Fin pour ;
16  Fin.
```

Une évaluation expérimentale sur des ensembles de données du monde réel et synthétiques montre que l'EMMA est plus efficace que le MINEPI.

2.3.4. L'algorithme TKE [18]

2.3.4.1. Définition

TKE (Top-k Frequent Episode mining) : Soit une fenêtre définie par l'utilisateur, un entier K > 0 et une séquence d'événements complexes. Le problème du l'extraction d'épisodes fréquents top-k consiste à trouver un ensemble de Top-k épisodes tel que leur support soit supérieur ou égal à celui de tout autre épisode non dans T.

- **Exemple :**

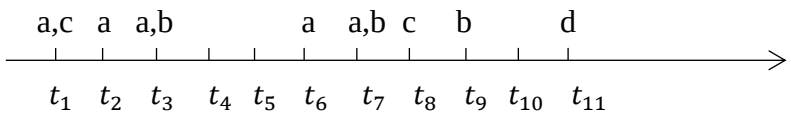


Figure 2-4 : une séquence complexe.[18]

Les épisodes fréquents trouvés dans la séquence de la figure 04 pour minsup=2 et winlen=2 sont ⟨{a,b},{a},{b}, ⟨{a},{a,b}, ⟨{a},{a}, ⟨{a}, ⟨{b},et ⟨{c}.Leurs valeurs de support sont 2, 2, 2, 3, 5, 3 et 2 respectivement.

Les Top-3 épisodes fréquents trouvés dans la séquence d'événement complexe de la figure 04 sont ⟨ {a}, {a}⟩ , ⟨{a}⟩ et ⟨{b}⟩ .

Il convient de noter que dans le cas où plusieurs épisodes ont le même support, le problème de l'extraction d'épisodes fréquents top-k peut avoir plusieurs solutions (différents ensembles d'épisodes peuvent former un ensemble d'épisodes top-k). De plus, pour certaines bases de données et valeurs de winlen, moins de k épisodes peuvent être trouvés.

2.3.4.2. Les phases de l'algorithme TKE

-TKE recherche les épisodes fréquents tout en conservant une liste des meilleurs épisodes actuels trouvés jusqu'à présent. TKE repose sur un minsup interne initialement défini sur 1, qui est ensuite progressivement augmenté à mesure que de nouveaux modèles sont trouvés. L'augmentation de minsup interne permet de réduire l'espace de recherche. Lorsque l'algorithme se termine, les épisodes top-k sont trouvés. L'algorithme TKE a trois paramètres d'entrée : une séquence d'entrée, k et winlen. La sortie est un ensemble d'épisodes fréquents top-k. TKE se compose de quatre phases, qui sont inspirées de l'algorithme EMMA mais adaptées pour une extraction efficace des épisodes top-k. EMMA a été choisi comme base pour TKE car il s'agit de l'un des algorithmes d'extraction d'épisodes les plus efficaces.

- **Phase 01 :** Trouver les top-k événements.

TKE définit d'abord minsup = 1. Ensuite, l'algorithme scanne la séquence d'entrée S une fois pour compter le support de chaque événement. Ensuite, le minsup est défini sur celui du k-ième événement le plus fréquent. L'augmentation du support à cette phase est une optimisation appelée augmentation d'épisode unique (SEI). Ensuite, TKE supprime tous les événements ayant un support inférieur à minsup de S, ou les ignore du traitement ultérieur. Ensuite, TKE analyse à nouveau la base de données pour créer une structure verticale appelée liste d'emplacements pour chaque événement (fréquent) restant. L'ensemble des événements fréquents est noté E′. La structure de la liste d'emplacements est définie comme suit :

La liste d'emplacement (Location List) d'un événement est notée locList(e) et définie comme la liste de toutes ses positions dans S. Le support d'un événement peut être dérivé de sa liste d'emplacement comme sup(e) =|locList(e)|.

Considérons la séquence d'entrée (Figure 04), S= ⟨({a,c},t1),({a},t2),({a,b},t3),({a},t6),({a, b},t7),({c},t8),({b},t9),({d},t11)⟩ , k= 3 et winlen= 2.Après le premier lecture, on trouve que le support des événements a,b,c et d sont 5, 3,2 et 1, respectivement. Ensuite, minsup est défini sur le support du k-ième événement le plus fréquent, c'est-à-dire minsup = 2. Après avoir supprimé les événements non fréquents, la séquence
S= ⟨({a,c},t1),({a},t2),({a,b},t3),({a},t6),({a,b},t7),({c},t8),({b},t9)⟩ Ensuite, les listes d'emplacement de a,b et c sont construites car leur support n'est pas inférieur à 2. Ces listes sont locList(a) = {1,2, 3,6,7}, locList(b) = {3,7,9} et locList(c) = {1,8}.

- **Phase 02 :** Trouver les Top-k épisodes parallèles.

La deuxième phase consiste à trouver les épisodes parallèles top-k en combinant les événements fréquents trouvés dans la phase 01. Ceci est mis en œuvre comme suit. Tout d'abord, tous les événements fréquents sont insérés dans un ensemble d'épisodes (PEpisodes). Ensuite, TKE essaie de joindre chaque épisode fréquent ep à chaque événement fréquent ∈ E′ | e ∉ ∧∀ f ∈ ep, f < e, pour générer un épisode parallèle plus grand newE = ep ∪{e}, appelé extension parallèle de ep. La liste des emplacements de newE est créée, qui est définie comme : locList(newE) ={p | p∈locList(e) ∧ ∃q∈locList(ep) | t(p) =t(q)}, où t(p) désigne l'horodatage correspondant à la position p dans S. Ensuite, si le support de newE n'est pas inférieur à minsup selon sa liste d'emplacement, alors newE est inséré dans PEpisodes, puis minsup est mis au support du k-ième épisode le plus fréquent dans les épisodes. Enfin, tous les épisodes des PEpisodes ayant un support inférieur à minsup sont supprimés. Ensuite, PEpisodes contient le top-k des épisodes parallèles fréquents.

Considérez l'exemple courant. L'extension parallèle de l'épisode ⟨{a}⟩ avec l'événement b est faite pour obtenir l'épisode ⟨{a,b}⟩ et sa liste d'emplacement locList(⟨{a,b}⟩) ={4,7}. Ainsi, sup(⟨{a,b}⟩) =|locList(⟨{a,b}⟩)| = 2. Ce processus est effectué pour générer d'autres extensions parallèles telles que ⟨ {a,c}⟩ et calculer leur support valeurs. Après la phase 2, les épisodes top-k parallèles sont ⟨{a}⟩ , ⟨{b}⟩ , ⟨{c} et ⟨{a,b}⟩ , avec des valeurs de support de 5, 3, 2, 2, respectivement, et minsup= 2.

- **Phase 03 :** Ré-encodage de la séquence d'entrée utilisant les épisodes parallèles.

La phase suivante consiste à ré-encoder la séquence d'entrée S à l'aide des k premiers épisodes parallèles pour obtenir une séquence ré-encodée S'. Ceci est fait en remplaçant les événements dans les séquences d'entrée par les épisodes parallèles top-k trouvés dans la phase 2. À cette fin, un identifiant unique est attribué à chaque épisode top-k.

Par exemple, les ID #1, #2, #3 et #4 sont assignés respectivement aux Top-k épisodes parallèles 〈{a}〉, 〈{b}〉, 〈{c}〉 et 〈〈{a,b}〉. Ensuite, la séquence d'entrée est ré-encodée comme :

S=〈({#1#3}, t1), ({#1}, t2), ({#1, #2, #4}, t3), ({#1}, t6), ({#1, #2, #4}, t7), ({#3}, t8), ({#2}, t9)〉.[18]

- **Phase 04 :** Trouver les Top-k épisodes composites.

Ensuite, l'algorithme TKE essaie de trouver les épisodes composites top-k. Tout d'abord, les épisodes parallèles top-k sont insérés dans un ensemble d'épisodes (CEpisodes). Ensuite, TKE joindre chaque épisode composite fréquent ep ∈ CEpisodes avec chaque événement fréquent ∈ PEpisodes pour générer un épisode plus grand appelé extension en série de ep par e. Formellement, l'extension sérielle d'un épisode ep = 〈SE1, SE2,..., SEx〉 avec un épisode parallèle donne l'extension sérielle d'épisode (ep,e) = 〈SE1, SE2,...,SEx, e〉. La liste liée (BoundList) de newE est créé.

Si le support de newE n'est pas inférieur à minsup selon sa liste liée (BoundList), alors newE est inséré dans CEpisodes, puis minsup est défini sur le support du k-ème épisode le plus fréquent dans Cepisodes. Enfin, tous les épisodes des Cepisodes ayant un support inférieur à minsup sont supprimés. À la fin de cette phase, Cepisodes contient le top-k des épisodes composites fréquents et l'algorithme se termine.

Par exemple, la liste liée de S=〈{a}, {a}〉 est boundList(〈{a}, {a}〉) = {[t1, t2], [t2, t3], [t6, t7]}. Ainsi, sup(〈{a}, {a}〉) =| {t1, t2, t6} |=3. Ensuite, TKE considère d'autres extensions série et calcule leurs valeurs de support de la même manière. Après la phase 4, minsup= 3 et les Top-k épisodes composites sont : 〈{a}〉, 〈{b}〉 et 〈{a}, {a}〉, avec un support de 5, 4 et 3 respectivement (le résultat final).

Algorithme 7 : TKE [18]

L'entrée : S : une séquence d'entrée, k : un nombre spécifié par l'utilisateur, winlen : la longueur de la fenêtre

La sortie : Le top-k des épisodes fréquents

```
Début
    minsup←1 ;
    Lire S pour calculer sup(e) pour chaque événemente∈E ;
    minsup←support de k-éme événement le plus fréquent∈E ;
    E′←{e|e∈E∧sup(e)≥minsup} ;
    Supprimer (ou ensuite ignorer) chaque événement ∉ E′ de S ;
    Lire S pour créer la liste d'emplacement de chaque événement fréquent ∈ E′ ;
    PEpisodes←E′ ;
    Pour chaque épisode parallèle ep ∈ PEpisodes tels que sup(ep)≥ minsup Faire
        Pour chaque évènement ∈ E′ tels que sup(e) ≥ minsup Faire
            newE←parallelExtension(ep, e); // et construire la liste des
            emplacements de newE.
            Si sup(newE) ≥ minsup Alors
                PEpisodes ←PEpisodeP Épisodes ∪{newE};
                minsup←support du k-ème épisodes les plus fréquents dans PEpisodes;
            Fin si ;
        Fin pour ;
    Fin pour ;
    Ré-encoder la séquence S en une séquence S′ en utilisant les épisodes parallèles ;
    CEpisodes←PEpisodes;
    Pour chaque épisode composite ep ∈ CEpisodes tels que sup(ep) ≥ minsup Faire
        Pour chaque évènement e ∈ PEpisodes tel que sup(e)≥ minsup Faire
            newE←serialExtension(ep, e); // et construire la liste liée de newE.
            Si sup(newE) ≥ minsup Alors
                CEpisodes←CEpisodes∪{newE};
                minsup←support du k-ième épisode le plus fréquent dans les CEpisodes;
            Fin si ;
        Fin pour ;
    Fin pour ;
    Retourner CEpisodes;
Fin.
```

2.4. Conclusion

Tout au long de ce chapitre, on a étudié la technique d'extraction d'épisode ainsi que quelques algorithmes proposés dans la littérature et disponible aux niveaux de la bibliothèque SPMF.

Le prochain chapitre sera consacré à notre contribution l'amélioration de l'algorithme (EMMA), puis quelques tests de comparaisons pour mesure les performances de notre proposition.

3.1. Introduction

Dans ce dernier chapitre après l'aperçu théorique général des chapitres précédents, nous avons appliqué notre objectif par un côté pratique, ce dernier est une amélioration de l'algorithme EMMA, et comparaison entres les algorithmes (EMMA et TKE) et (EMMA et E-EMMA), note que notre comparaison est entre deux algorithme dans l'extraction d'épisodes fréquents, et montres des résultats grâce à des courbes graphiques.

3.2. Présentation des outils de développement

3.2.1. NetBeans [19]

Est un environnement développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Développent and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refacturation, éditeur graphique d'interfaces et de pages Web).

Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développent Kit JDK est requis pour les développements en Java. NetBeans constitue par ailleurs une plate-forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate-forme.

3.2.2. Java [20]

Java est un langage de programmation et une plate-forme informatique crées par Sun Microsystems en 1995. Il s'agit de la technologie sous-jacente qui permet l'exécution de programmes modernes et performants, notamment dans la construction des utilitaires, des jeux et des applications professionnelles. Java est utilisée sur plus de 850 millions d'ordinateurs de bureau et plus d'un milliard de périphériques dans le monde, dont des périphériques mobiles et des systèmes de diffusion télévisuelle.

3.2.3. Une bibliothèque de donnée extra source (SPMF) [8]

SPMF est une bibliothèque minière exploration de données open-source écrit en Java, spécialisé dans le secteur minier de modèle. Il est distribué sous licence GPL. Il propose des implémentations de 210 algorithmes d'extraction des données il code source de chaque algorithme peut être facilement intégré dans d'autres logiciels Java.SPMF peut être utilisé comme un programme autonome avec une interface utilisateur simple ou à partir de la ligne de commande .De plus, le code source de chaque algorithme peut être facilement intégré dans d'autres logiciels Java .En outre, certains wrappers non officiels sont disponibles pour d'autres langages tels que Python, R et Weka.SPMF est rapide et léger (aucune dépendance à d'autres bibliothèques).La version actuelle est la v2.47 et a été publiée le 28 mai 2021.

3.3. Description de la base de données

Une base de données de séquence est un ensemble de séquences où chaque séquence est une liste d'itemsets. Un jeu d'éléments est un ensemble non ordonné d'éléments distincts.[21]

A titre d'exemple, prenons le tableau 05 qui contient une séquence composée de 11 points de temps (t1, t2, …, t10) et de 4 types d'événements (1, 2, 3, 4) :
Chaque ligne de la séquence est appelée un ensemble d'événements et se compose de :

- Un point dans le temps (première colonne)
- Un ensemble d'événements (deuxième colonne)

Point de temps	Evènement
t1	1
t2	1
t3	1 2
t6	1
t7	1 2
t8	3
t9	2
t1	4

Tableau 3-1 : Base de données de séquence. [22]

Il est clair qu'à l'instant 1, l'événement 1 s'est produit. Puis au point de temps 2, l'événement 1 s'est produit à nouveau. Puis a l'instant t3, les événements 1 et 2 se sont produits simultanément. Etc.

Le format de fichier d'entrée utilisé dans les algorithmes (EMMA et TKE), est un fichier texte. Un élément (événement) est représenté par un entier positif. Une transaction (ensemble d'événements) est une ligne dans le fichier texte. Dans chaque ligne (ensemble d'événements), les éléments sont séparés par un seul espace. On suppose que tous les éléments (événements) d'une même ligne de transaction) sont triés selon un ordre total (par exemple, ordre croissant) et qu'aucun élément ne peut apparaître deux fois dans le même ensemble d'événements. Chaque ligne est éventuellement suivie du caractère "|" puis l'horodatage de l'événement défini (ligne). Notez qu'il est possible d'exécuter (EMMA et TKE) sur une base de données qui n'a pas d'horodatage. Dans ce cas, le paramètre booléen (le troisième paramètre) doit être défini sur " true "pour indiquer que l'ensemble de données n'a pas d'horodatage. S'il n'y a pas d'horodatage et que le paramètre est défini sur " true", l'algorithme supposera que chaque ligne a un horodatage qui augmente de 1. [22]

Nous avons pris deux bases de données qui sont :

1) segmentation de la peau : et un ensemble de données qui est transformé à partir de l'ensemble de données de la segmentation de la peau, obtenu à partir de https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation Converti par Zhang Zhongjie. [23], (le contenu est présenté par le tableau 3-1).

- **Informations sur l'ensemble de données [24]**

L'ensemble de données sur la peau est collecté en échantillonnant au hasard les valeurs B, G, R à partir d'images de visage de divers groupes d'âge (jeune, moyen et vieux), de groupes ethniques (blanc, noir et asiatique) et de sexes obtenus à partir de la base de données FERET et de la base de données PAL... La taille totale de l'échantillon d'apprentissage est de 245057 ; dont 50859 sont des échantillons de peau et 194198 sont des échantillons non cutanés. Base de données d'images couleur FERET, Base de données de visages PAL du « Productive Aging Laboratory », Université du Texas à Dallas.

- **Informations sur les attributs [24]**

Cet ensemble de données est de la dimension 245057 * 4 où les trois premières colonnes sont les valeurs B, G, R (caractéristiques x1, x2 et x3) et la quatrième colonne est celle des étiquettes de classe (variable de décision y).

Figure 3-1 : Présenter le premier fichier d'entrée.

2) enregistrements des données individuelles : Cet ensemble de données est transformé à partir de http://archive.ics.uci.edu/ml/datasets/Record+Linkage+Comparison+Patterns .
Converti par Zhang Zhongjie, (le contenu est présenté par la Figure 3-1).

- **Informations sur l'ensemble de données [25]**

Les enregistrements représentent des données individuelles comprenant le prénom et le nom de famille, le sexe, la date de naissance et le code postal, qui ont été recueillies par insertions itératives au cours de plusieurs années. Les modèles de comparaison dans cet ensemble de données sont basés sur un échantillon de 100 000 enregistrements. Les paires de données ont été classées comme « correspondantes » ou « non correspondantes » lors d'un examen manuel approfondi impliquant plusieurs documents.

- **Informations sur les attributs [25]**

1. id_1 : identifiant interne du premier enregistrement.
2. id_2 : identifiant interne du deuxième enregistrement.
3. cmp_fname_c1 : accord du prénom, premier élément
4. cmp_fname_c2 : accord du prénom, deuxième composante
5. cmp_lname_c1 : accord du nom de famille, premier élément
6. cmp_lname_c2 : accord du nom de famille, deuxième composante
7. cmp_sex : accord sexe
8. cmp_bd : accord de date de naissance, composante jour
9. cmp_bm : accord de date de naissance, composante mois
10. cmp_by : accord de date de naissance, composante année
11. cmp_plz : accord de code postal
12. is_match : état correspondant (TRUE pour les correspondances, FALSE pour les non-correspondances).

Figure 3-2 : Présenter le deuxième fichier d'entrée.

3.4. Les algorithmes utilisés

Comme on a déjà présenté les deux algorithmes EMMA et TKE dans le chapitre 2 dans la section (**2.3.3.**) et (**2.3.4.**) respectivement.

On passe maintenant à une présentation sous forme d'organigramme pour mieux comprendre les différents types de ces algorithmes.

3.4.1. Algorithme EMMA

Cet algorithme peut être vu comme trois phases quant se cité et comme suit :

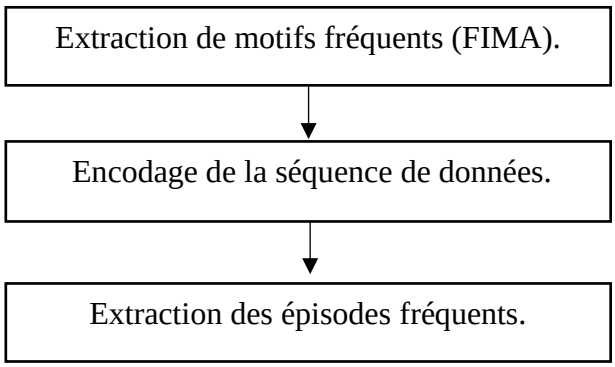


Figure 3-3 : Les phases de l'algorithme EMMA.

3.4.1.1. Extraction de motifs fréquents (FIMA)

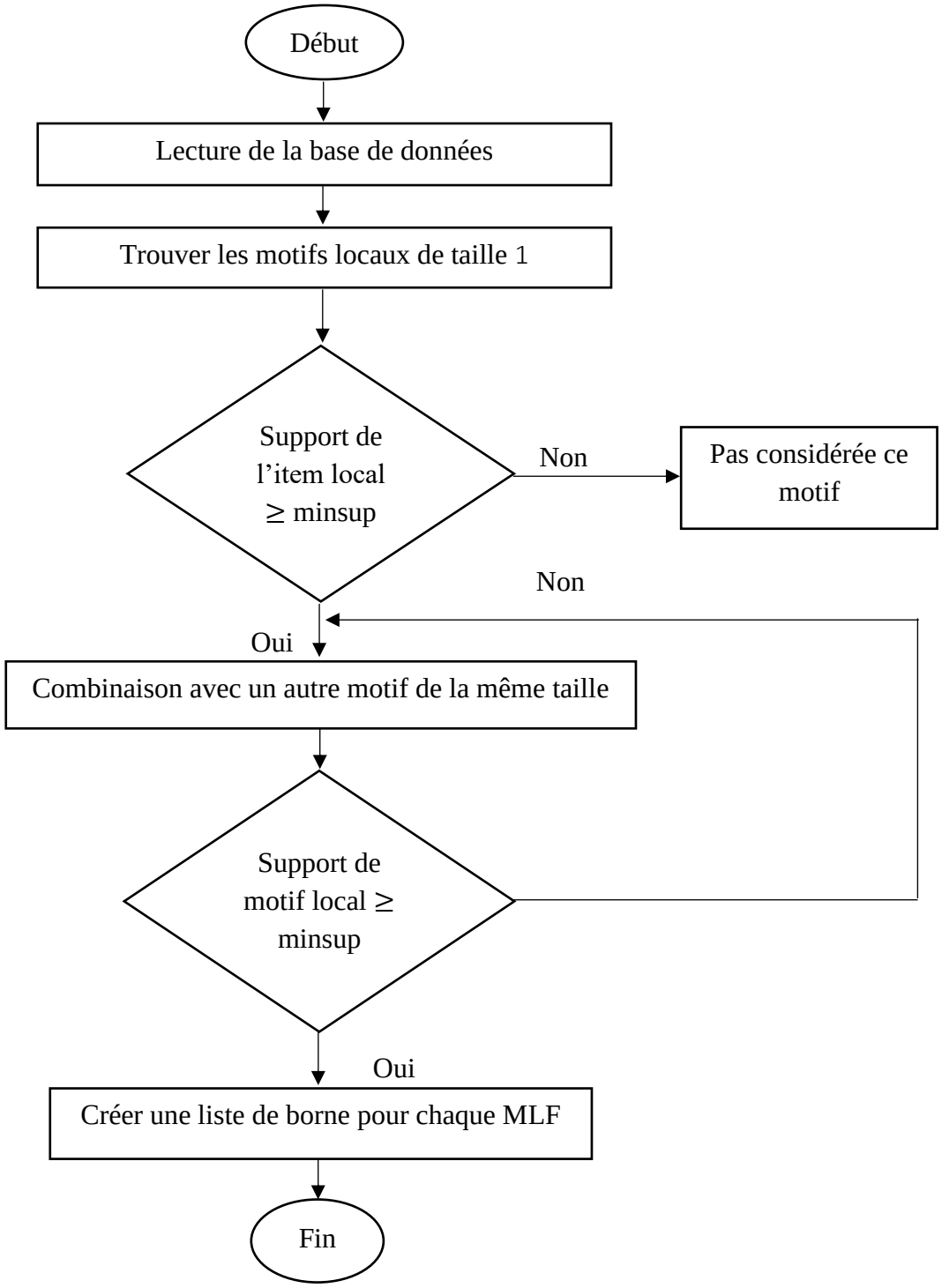


Figure 3-4 : Organigramme de la phase d'extraction de motifs fréquents (algorithme FIMA).

3.4.1.2. L’encodage de la séquence de données :

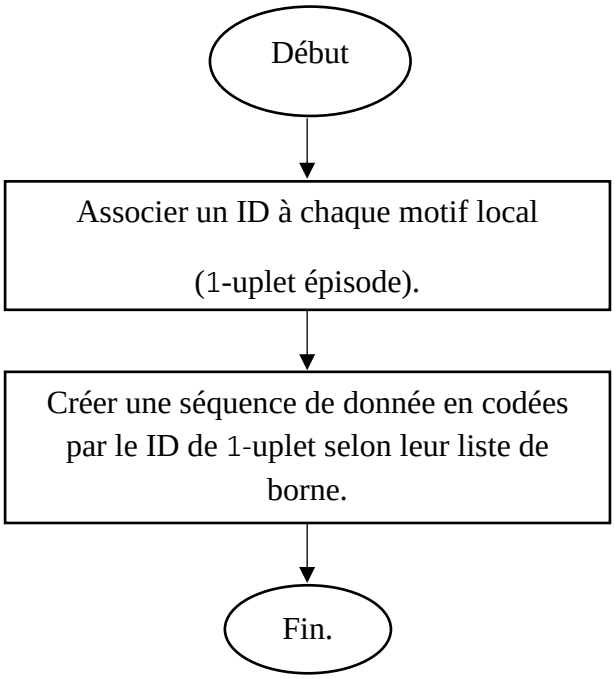


Figure 3-5 : Organigramme de la phase de l’encodage de la séquence de données.

3.4.1.3. L'extraction des épisodes fréquent « utilise le SDD en codées ».

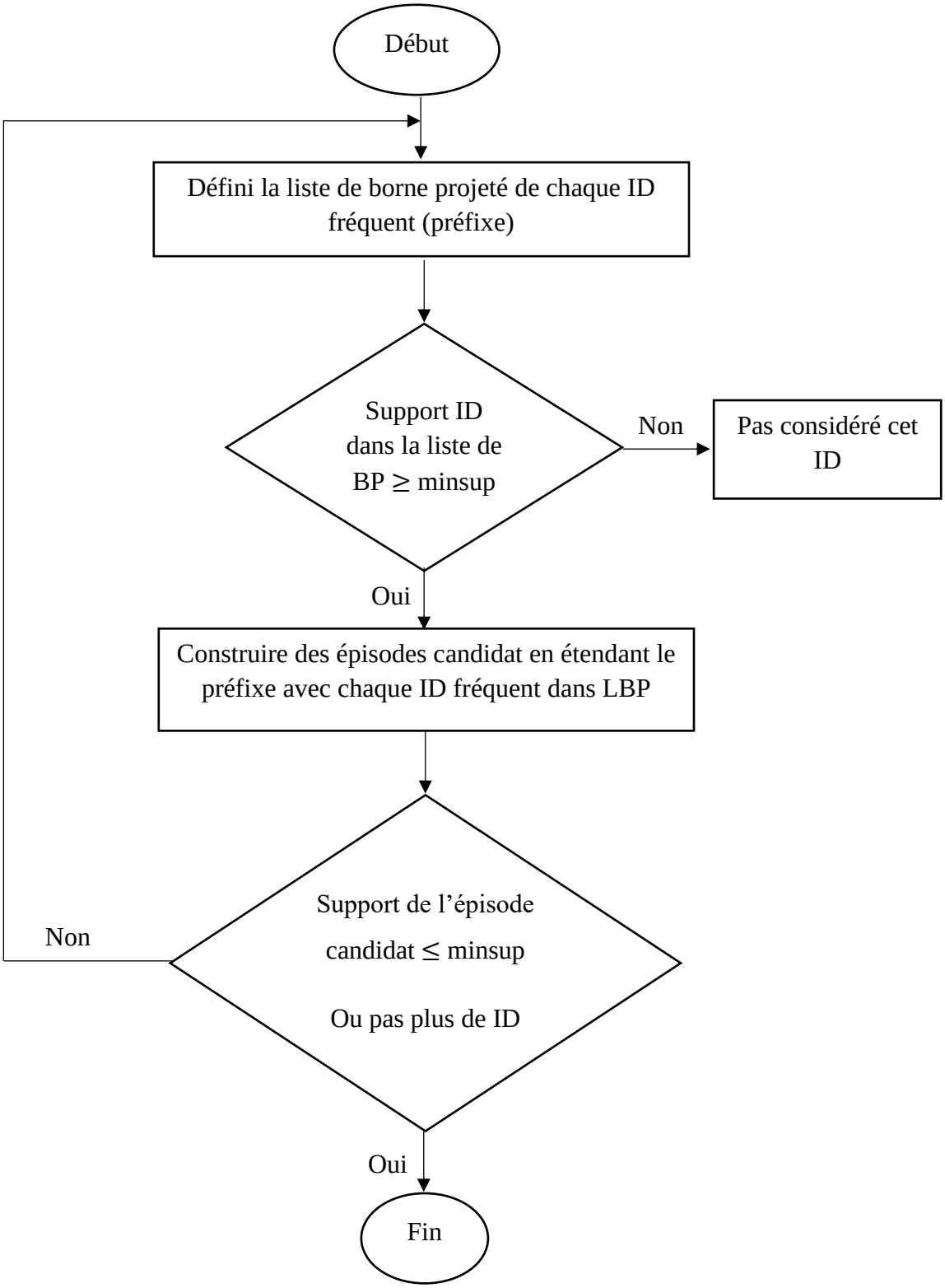


Figure 3-6 : Organigramme de la phase de l'extraction des épisodes fréquent.

3.4.2. Algorithme TKE

Cet algorithme peut être vu comme quatre phases quant se cité et comme suit :

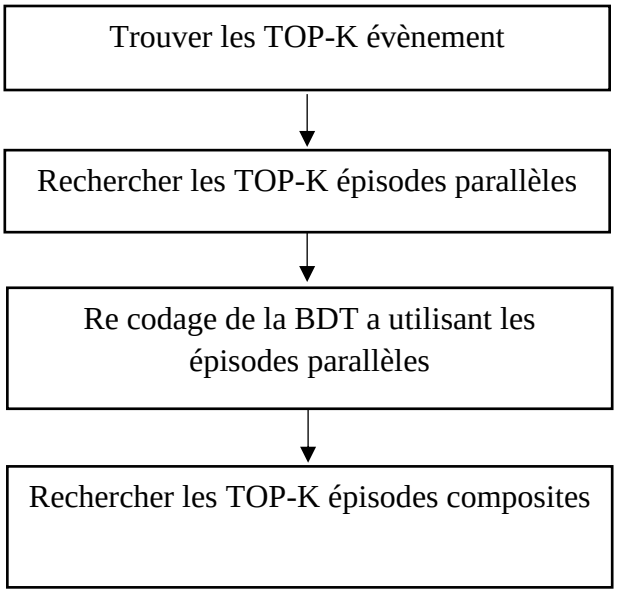


Figure 3-7 : Les phases de l'algorithme TKE.

3.4.2.1. Trouver les TOP-K évènement

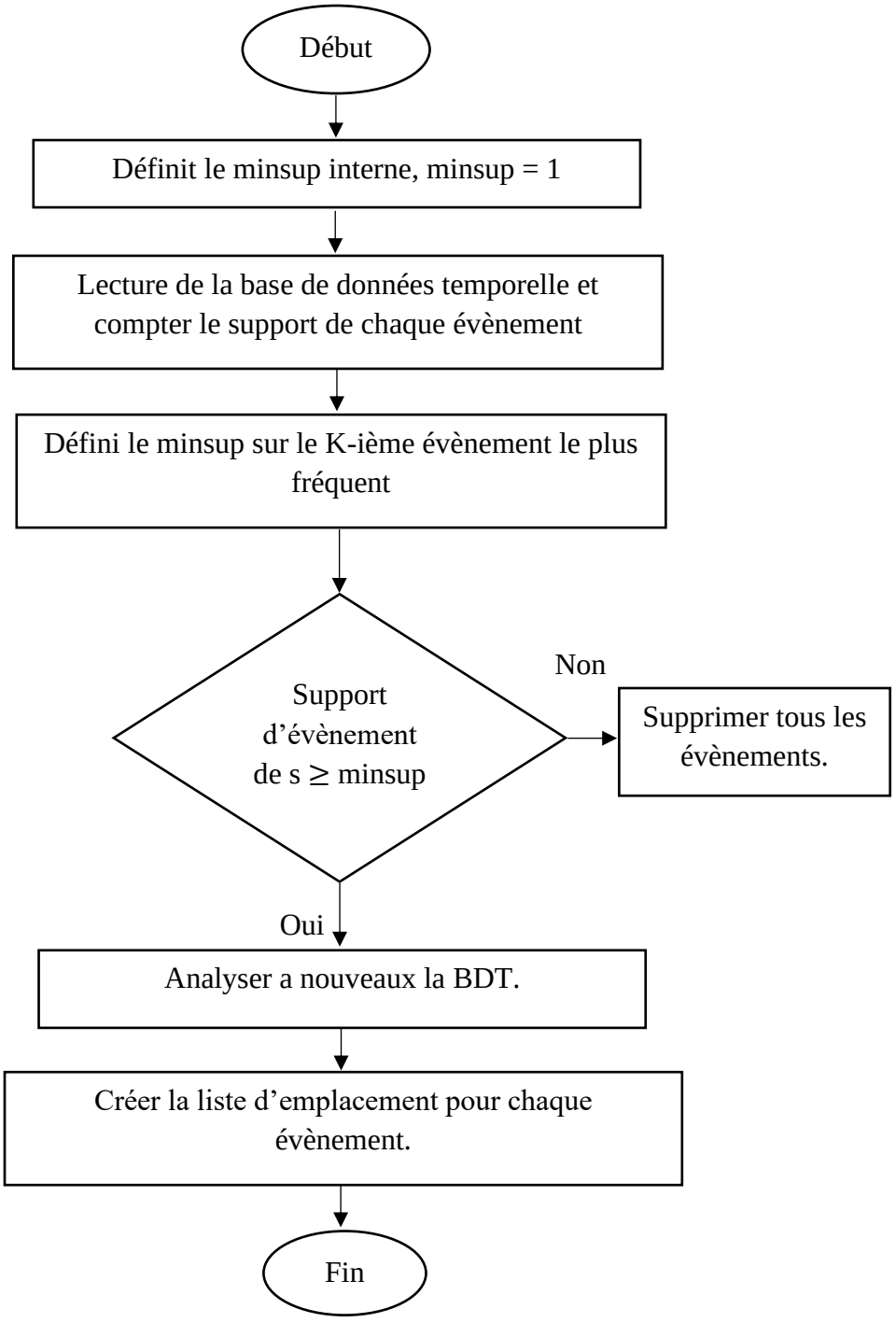


Figure 3-8 : Organigramme de la phase de trouver les TOP-K évènement**.**

3.4.2.2. Rechercher les TOP-K épisodes parallèles

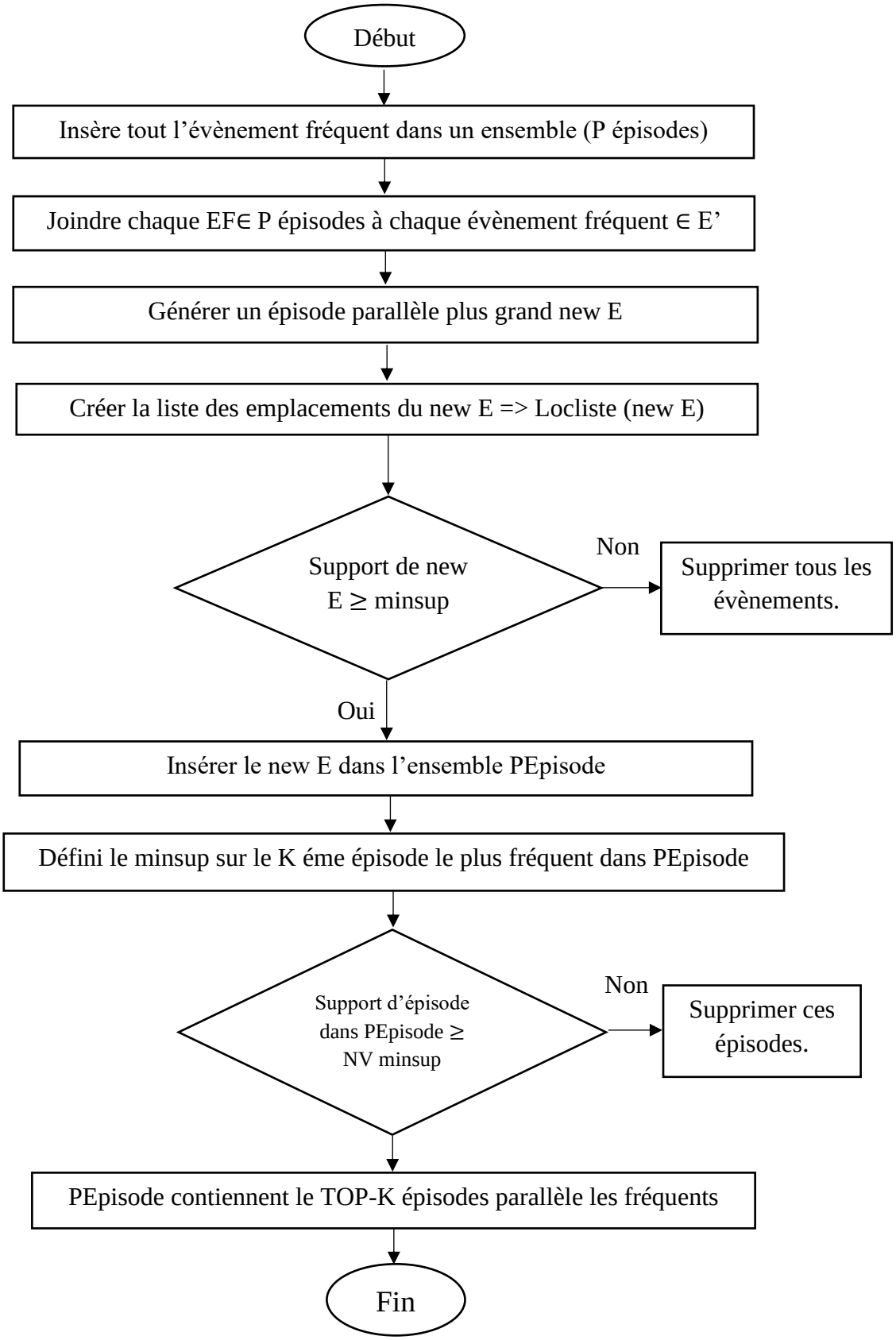


Figure 3-9 : Organigramme de la phase Rechercher les TOP-K épisodes parallèles.

3.4.2.3. Re codage de la BDT a utilisant les épisodes parallèles

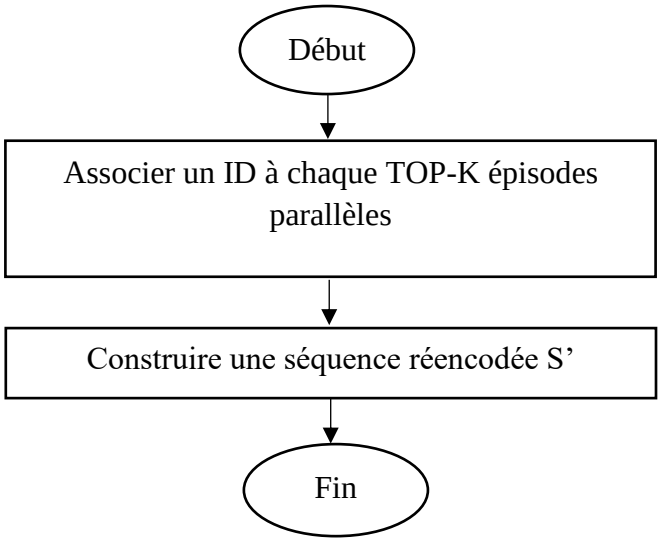


Figure 3-10 : Organigramme de la phase de Re-codage de la BDT a utilisant les épisodes parallèles.

3.4.2.4. Rechercher les TOP-K épisodes composites

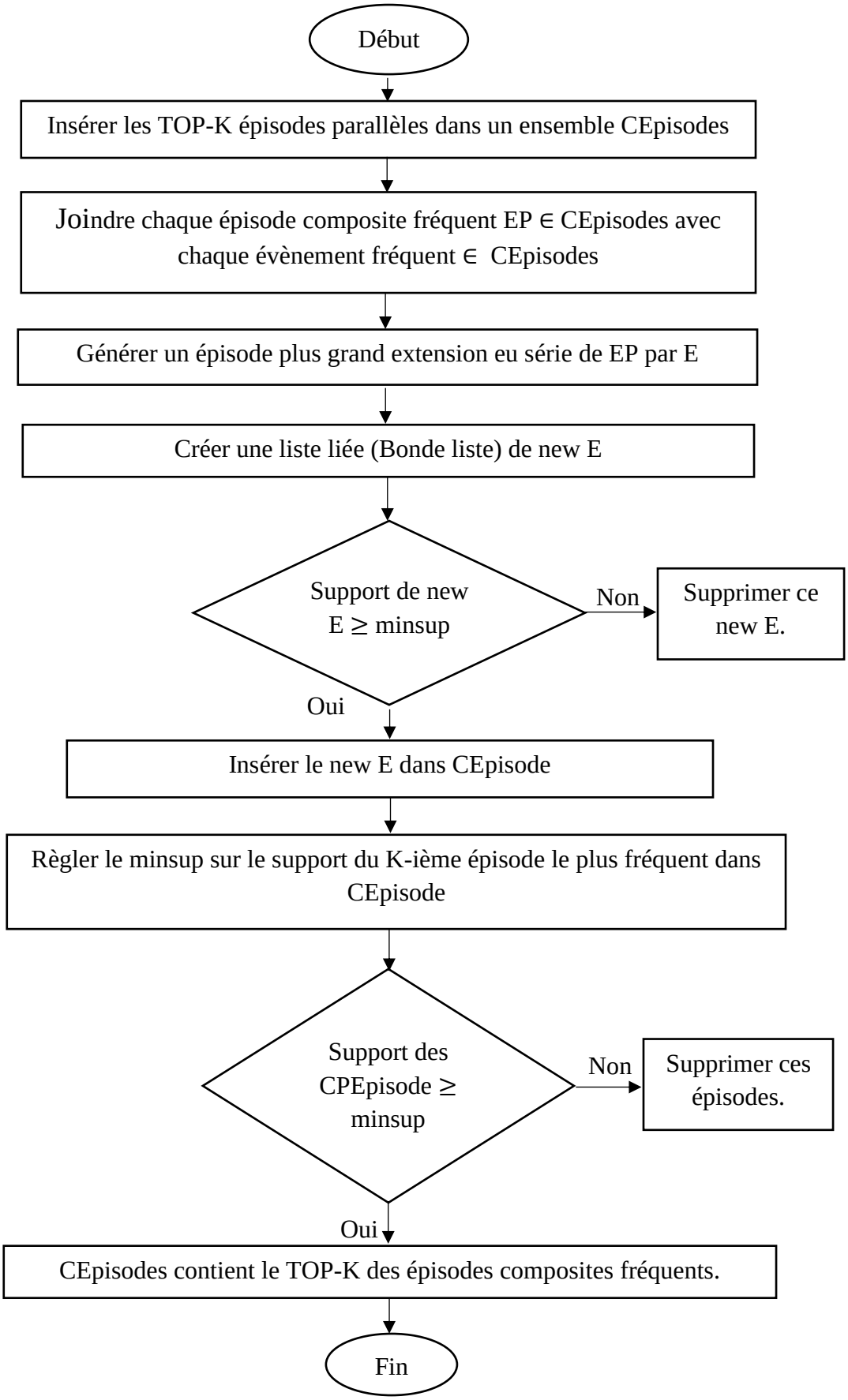


Figure 3-11 : Organigramme de la phase d Rechercher les TOP-K épisodes composites.

3.5. Résultats de l'exécution de l'algorithme EMMA et TKE

Minsup = 4

Base de données	Taille de la base	Algorithmes	Le nombre des candidats	Le nombre des épisodes fréquents	Temps total	Utilisation maximale de la mémoire en méga byte
Segmentation de la peau	T1 =245057 séquences	EMMA	52670	229	31 s	228.41957092285156 mb
		TKE	52887	229	30 s	227.69623565673020 mb
	T2 = 10000 séquences	EMMA	34410	185	7 s	97.69385583496094 mb
		TKE	34583	185	6 s	97.86336517333984 mb
Enregistrements Des données individuelles	T1 = 8000 séquences	EMMA	23653632	4863	4 min et 27 s	101.32328186035156 mb
		TKE	23659668	4863	4 min et 8 s	102.44910430908203 mb
	T2 = 1500 séquences	EMMA	16642320	4079	2 min et 56 s	82.42758178710938 mb
		TKE	16647616	4079	2 min et 37 s	82.56795501708984 mb

Tableau 3-2 : Résultats de l'exécution des algorithmes EMMA et TKE avec le minsup = 4.

Minsup = 6

Base de données	Taille de la base	Algorithme	Le nombre des candidats	Le nombre des épisodes fréquents	Temps total	Utilisation maximale de la mémoire en méga byte
Segmentation de la peau	T1= 245057	EMMA	52670	229	29 s	228.39661407470703 mb
		TKE	52887	229	28 s	228.82666778564453 mb
	T2= 10000	EMMA	31506	177	6 s	97.78911590576172 mb
		TKE	31679	177	7 s	98.46469116210938 mb
Enregistrements Des données individuelles 2	T1= 8000	EMMA	13656720	3695	3 min et 36 s	101.1467269897461 mb
		TKE	13661749	3695	3 min et 20 s	101.34571838378906 mb
	T2= 1500	EMMA	11178992	3343	2 min et 28 s	81.4457778930664 mb
		TKE	11185360	3343	2 min et 20 s	81.49578094482422 mb

Tableau 3-3 : Résultats de l'exécution des algorithmes EMMA et TKE avec le minsup =6.

Minsup = 9

Base de données	Taille de la base	Algorithme	Le nombre des candidats	Le nombre des épisodes fréquents	Temps total	Utilisation maximale de la mémoire en méga byte
Segmentation de la peau	T1= 245057	EMMA	51756	227	33 s	221.7746124267578 mb
		TKE	51973	227	30 s	230.00342559814453 mb
	T2= 10000	EMMA	31506	177	10 s	97.97395324707031 mb
		TKE	31679	177	9 s	97.01362609863281 mb
Enregistrement des données individuelles	T1= 8000	EMMA	9434112	3071	3 min et 1 s	100.08096313476562 mb
		TKE	9438429	3071	2 min et 50 s	100.21383666992188 mb
	T2= 1500	EMMA	9434112	3071	2 min et 18 s	81.49519348144531 mb
		TKE	9438371	3071	2 min et 9 s	81.50331115722656 mb

Tableau 3-4 : Résultats de l'exécution des algorithmes EMMA et TKE avec le minsup = 9.

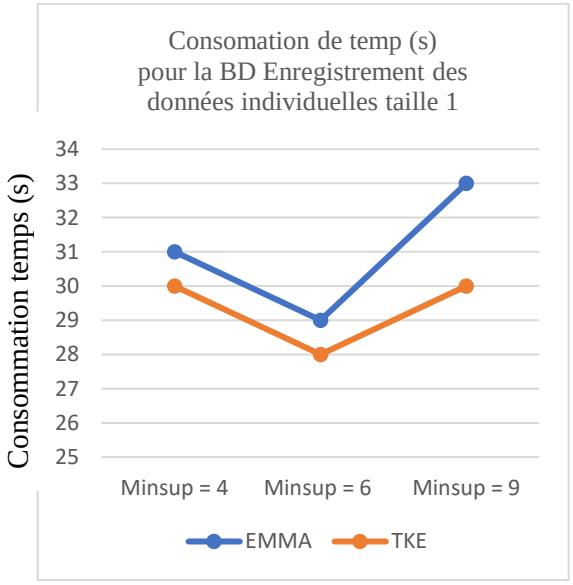


Figure 3-12 : Résultats obtenus de consommation consommation de temps pour minsup (4,6,9).

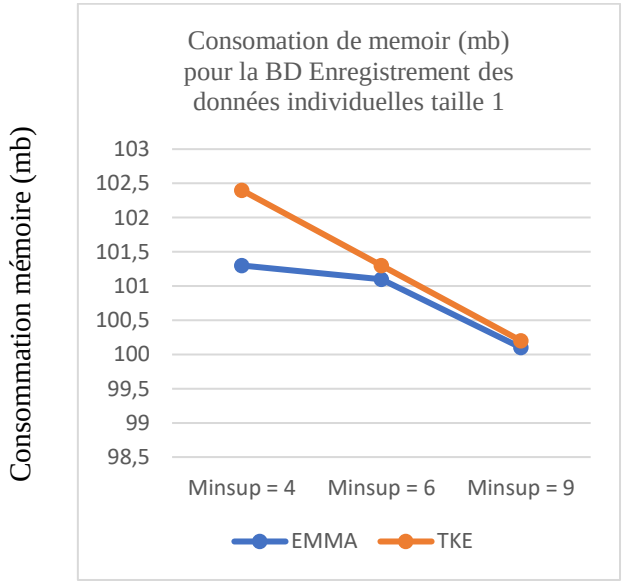


Figure 3-13 : Résultats obtenus de de mémoire pour minsup (4,6,9).

Discussion

En se basant sur les résultats précédents, on peut remarquer la différence entre les deux algorithmes EMMA et TKE avec les minsups suivants : 4, 6 et 9

Nous notons qu'en général, Les résultats étaient presque égaux en termes de temps d'exécution et d'espace mémoire, mais dans certains cas l'algorithme TKE était moins couteuse que Emma et c'était moins de 1 seconde pour le temps et moins de 1 mb pour la mémoire.

3.6. Résultats de l'exécution de l'algorithme EMMA, E-EMMA et TKE

Dans notre étude de l'algorithme EMMA, qui se concentre sur les épisodes sériels, nous y avons apporté quelques modifications pour que cet algorithme serve des épisodes parallèles où l'ordre n'est pas important, ce qui est utilisé dans de nombreux domaines comme la médecine (diagnostic des maladies).

De là, nous avons obtenu des résultats pour les algorithmes EMMA, TKE et E-EMMA « Enhanced-EMMA », donc les résultats d'exécution étaient comme indiqué dans les tableaux suivants :

Minsup = 4

Bas de données	Taille de la base	Algorithme	Le nombre des candidats	Le nombre des épisodes fréquents	Temps total	Utilisation maximale de la mémoire en méga byte
Enregistrement des données individuelles	T1= 4000	E-EMMA	481569	18047	36 min et 42 s	26.752639770507812 mb
	T2= 1500	E-EMMA	75122	4079	1 min et 15 s	27.571701049804688 mb

Tableau 3-5 : Résultats de l'exécution de l'algorithmes E-EMMA avec le minsup = 4.

Minsup = 6

Bas de données	Taille de la base	Algorithme	Le nombre des candidats	Le nombre des épisodes fréquents	Temps total	Utilisation maximale de la mémoire en méga byte
Enregistrement des données individuelles	T1= 4000	E-EMMA	388904	15077	36 min et 48 s	25.84630584716797 mb
	T2= 1500	E-EMMA	58068	3343	1 min et 15 s	27.615692138671875mb

Tableau 3-6 : Résultats de l'exécution de l'algorithmes E-EMMA avec le minsup = 6.

Minsup = 9

Bas de données	Taille de la base	Algorithme	Le nombre des candidats	Le nombre des épisodes fréquents	Temps total	Utilisation maximale de la mémoire en méga byte
Enregistrement des données individuelles	T1= 4000	E-EMMA	340559	13695	36 min et 49 s	26.38219451904297 mb
	T2= 1500	E-EMMA	46760	3071	1 min et 16 s	27.075332641601562 mb

Tableau 3-7 : Résultats de l'exécution de l'algorithmes E-EMMA avec le minsup = 9.

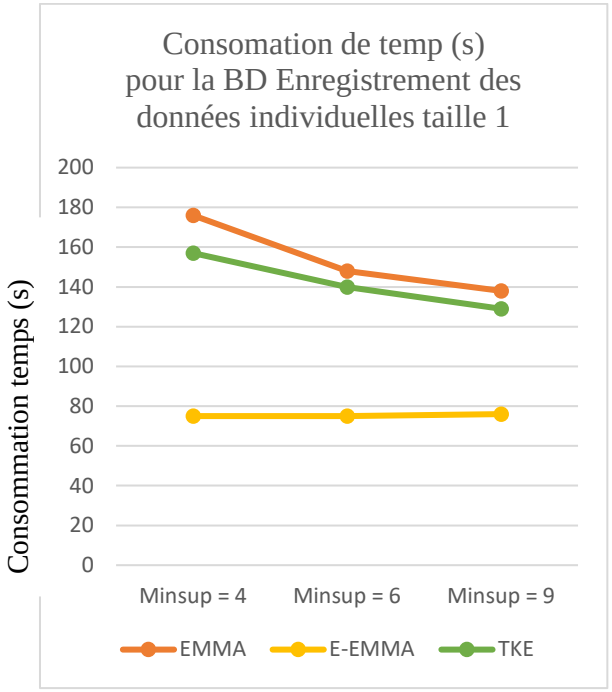


Figure 3-14 : Résultats obtenus de consommation consommation de temps pour minsup (4,6,9).

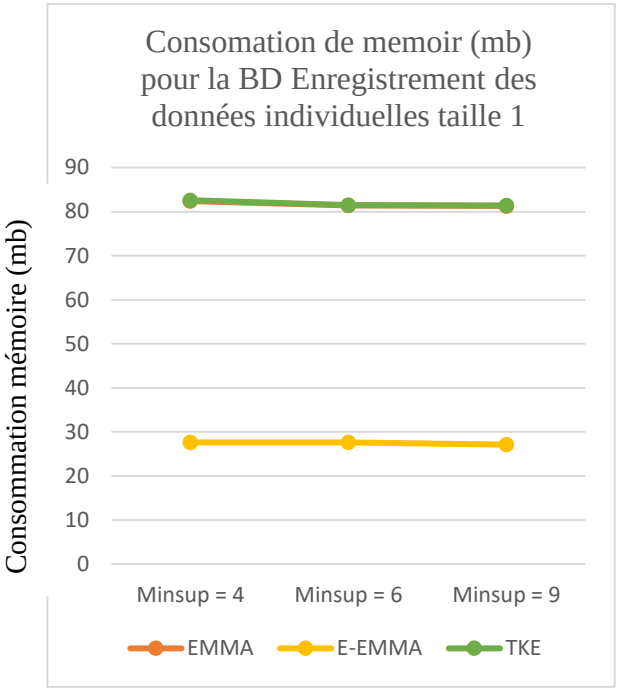


Figure 3-15 : Résultats obtenus de de mémoire pour minsup (4,6,9).

Discussion

En se basant sur les résultats précédents avec les minsups suivants : 4,6 et 9, la différence réside dans :

Nous avons trouvé que E-EMMA est moins cher que EMMA et TKE en termes d'espace mémoire et de temps d'exécution, car de nombreux épisodes qui ont les même itemsets sont considérés comme un seul épisode car l'ordre de ces itemsets n'est pas important dans ce cas.

3.7. Conclusion

Dans ce chapitre nous avons montré les outils utilisés pour l'amélioration, après nous avons fait les organigrammes des deux algorithmes étudier et nous avons choisi deux bases de données nous appliquons ces algorithmes, En fin nous introduisons tous les résultats obtenus après l'exécution de plusieurs fois de chaque algorithme et comparer entre ces algorithmes.

Conclusion générale :

A la fin de ce travail de fin d'étude nous avons cherché et découvert un nouveau domaine est la fouille de donnée, et nous avons bien comprendre les problématiques liées aux l'extraction des motifs séquentiels et l'extraction des épisodes fréquents.

Dans ce cadre de travail, nous avons étudié et essayé de comprendre et évaluer les performances de quelques algorithmes d'extraction des épisodes fréquents en termes de temps d'exécution et consommation de l'espace mémoire, et pour atteindre notre objectif (étude comparative), nous avons étudié deux algorithmes qui sont EMMA et TKE et obtenus des résultats qui nous a permis de connaitre la déférence entre ces deux algorithmes.

Nous pouvons conclure que l'extraction d'épisodes fréquents est une tâche d'exploration de données populaire pour analyser une séquence d'événements. Elle consiste à identifier toutes les sous-séquences d'événements qui apparaissent au moins pendant les minutes de disponibilité.

Nous avons remarqué à travers de cette étude que les algorithmes traditionnels ont un problème au niveau du paramètre minsup, car chaque fois que ce paramètre est très haut, les algorithmes peuvent trouver peu des motifs, il est donc possible de perdre des informations très importantes pour l'utilisateur. Si ce paramètre est trop bas, l'exécution prend beaucoup de temps et un espace de stockage élevée, sachant que l'utilisateur ne dispose généralement pas de beaucoup de temps et d'un espace de stockage pour analyser ses données.

Cela aurait été bien si nous prenions différents types de séquence de données pour faire notre étude, mais à la fin nous n'en avons pris que deux et avons pris deux tailles différentes de chacun.

Référence

[1] http://blog.naseej.com

[2] http://webia.lip6.fr/~rifqi/COURS2001-2002/IA/Clustering.pdf.

[3] http://e-biblio.univ-osta.dz/bitstream/handle/123456789/5633/MINF141.pdf?sequence=1&isAllowed=y

[4] http://webia.lip6.fr/~rifqi/COURS2001-2002/IA/Clustering.pdf.

[5] https://www.sciencedirect.com/science/article/abs/pii/S0020025520302103

[6] https://www.philippe-fournier-viger.com/Survey_high_utility_itemset2019_draft.pdf

[7] https://www.sciencedirect.com/science/article/abs/pii/S0957417416305735#keys0001

[8] https://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php

[9] JESSICA LIN, Experiencing SAX : a Novel Symbolic Representation of Time Series

[10] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203345-clustering-definition/

[11] Lina fahed,Prédire et influencer l'apparition des évènements dans une séquence complexe,2016.

[12] Adel Boubaker Hidri, Ahmed Selmi, Minyar Sassi Hidri, Discovery of Frequent Patterns of EpisodesWithin a Time Window for AlarmManagement Systems, janvier 2012.

[13] Lina Fahed, Armelle Brun, Anne Boyer, Prédiction au plus tôt d'événements par règles d'épisodes, janvier 2014

[14] Https://www.philippe-fournier-viger.com/spmf/MINEPI.php.

[15] https://www.philippe-fournier-viger.com/spmf/TUP.PHP

[16] Sonam Rathore; Siddharth Dawar ;Vikram Goyal ;Dhaval Patel

[17] Chia-Hui Chang, Efficient mining of frequent episodes from complex sequences, 20 July 2016

[18] Philippe Fournier-Viger, Yanjun Yang, Peng Yang, Jerry Chun-Wei Lin, et Unil Yun , TKE: Mining Top-K Frequent Episodes, 2020

[19] https://fr.wikipedia.org/wiki/NetBeans

[20] https://www.java.com/fr/download/help/whatis_java.html

[21] Bentoumi ISmail, Extraction de motifs séquentiels, 2017.

[22] Https ://www.philippe-fournier-viger.com/spmf/EMMA.php.

[23] https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

[24] https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation

[25] http://archive.ics.uci.edu/ml/datasets/Record+Linkage+Comparison+Patterns

ملخص:

ازدادت البيانات بشكل ملحوظ مؤخرًا وأفضل مثال على ذلك هو وسائل التواصل الاجتماعي، والتنقيب عن البيانات هو مجال يعتمد بشكل كبير على إيجاد طرق منهجية لتحليل هذا الكم الهائل من البيانات، ومن التقنيات في هذا المجال استخراج الحلقات المتكررة، وهي الأكثر شيوعًا للعثور على أنماط في تسلسل البيانات.

هناك العديد من الخوارزميات لاستخراج مثل هذه الأنماط (الحلقات) التي تظهر في تسلسل الاحداث والتي تعتمد بشكل أساسي على تعريف الحد الأدنى للدعم.

في هذا المشروع، طبقنا خوارزميات استخراج حلقات متكررة على قاعدتي بيانات مختلفتين لتقييم أدائها.

Abstract :

Data has increased very noticeably recently and the best example is social media, data mining is an area that relies heavily on finding methodical ways to analyze this huge amount of data, and one of the techniques in this domain is the extraction of frequent episodes, which are the most common to find patterns in sequences of data.

There are many algorithms for extracting such patterns (episodes) which appear in the sequence of events and which mainly rely on the definition of minsup (minimum support).

In this project, we applied frequent episode extraction algorithms on two different databases to assess their performance.

Résumé :

Les données ont récemment augmenté de manière très notable et le meilleur exemple est les réseaux sociaux, la fouille de données est un domaine qui repose fortement sur la recherche de moyens méthodiques pour analyser cette énorme quantité de données, et l'une des techniques dans ce domaine est l'extraction d'épisodes fréquents, qui sont les plus courants pour trouver des motifs dans des séquences de données.

Il existe de nombreux algorithmes permettant l'extraction de tels motifs (épisodes) qui apparaissent dans la séquence d'événements et qui reposent principalement sur la définition de minsup (support minimum).

Dans ce projet, nous avons appliqué des algorithmes d'extraction d'épisodes fréquents sur deux base données différentes pour évaluer leurs performances.