# Extraction de séquences fréquentes avec intervalles d'incertitude

Asma Ben Zakour*,**, Sofian Maabout*, Mohamed Mosbah*, Marc Sistiaga**

*LaBRI, Université Bordeaux 1, CNRS, FRANCE
maabout@labri.fr, mosbah@labri.fr
**2MoRO Solutions, Bidart, FRANCE
marc.sistiaga@2moro.fr, asma.ben-zakour@2moro.fr

**Résumé.** Lors de l'extraction des séquences, la granularité temporelle est plus ou moins importante selon les besoins des utilisateurs et les contraintes du domaine d'application. Nous proposons un algorithme d'extraction de séquences fréquentes par intervalles à partir de séquences à estampilles temporelles discrètes. Nous intégrons une relaxation des contraintes temporelles en introduisant la définition de "séquences temporelles par intervalles" (STI). Ces intervalles reflètent une incertitude sur les occurrences précises des évènements. Nous formalisons ce nouveau concept en exhibant certaines de ses propriétés et nous menons quelques expériences afin de comparer (qualitativement) nos résultats avec une autre proposition assez proche de la nôtre.

## 1 Introduction

L'extraction de séquences fréquentes (ESF) a été introduite par Agrawal et Srikant (1995) comme une extension de leur algorithme *A Priori* Agrawal et Srikant (1994) qui calcule les ensembles fréquents. Lors de l'ESF, la chronologie d'occurrence des événements est plus ou moins importante selon la nature des connaissances à extraire. En effet, il est parfois nécessaire de relaxer les contraintes de chronologie afin d'extraire des informations utiles. Dans cet article, nous proposons de *fusionner* des évènements consécutifs et proches temporellement en un ensemble d'évènements simultanés dont l'estampille temporelle n'est plus discrète mais exprimée par un intervalle temporel. Cet intervalle représente une incertitude sur les instants exacts des occurrences des évènements regroupés. Ce regroupement est contraint par la taille de la fenêtre glissante fixée par l'utilisateur. Ce travail est réalisé dans le cadre d'un projet industriel. Il s'agit d'anticiper des opérations de maintenance des équipements aéronautiques. Par exemple, considérons un historique d'utilisation d'avions. $V_i$ désigne le vol $i$ et $M_j$ désigne l'opération de maintenance $j$. Soit $\mathcal{S} = \{S_1, S_2\}$ un ensemble de séquences avec : $S_1 = \langle(0, V_1)(1, V_2)(2, V_3)(5, M_1)\rangle$ et $S_2 = \langle(0, V_1)(1, V_3)(2, V_2)(6, M_1)\rangle$. En fixant un support minimal à 2 et une taille de fenêtre glissante égale à 1, notre approche retourne la séquence : $\langle([0, 0]V_1)([1, 2]V_2\ V_3)([5, 6]M_1)\rangle$ qui traduit les faits suivants : « Lorsque le vol $V_1$ est effectué, $V_2$ et $V_3$ ont lieu *dans n'importe quel ordre* entre une et deux unités de temps après $V_1$, *i.e.*, dans l'intervalle $[1, 2]$. Par la suite, la réparation $M_1$ est effectuée dans l'inter-

valle $[5, 6]$ après le vol $V_1$ ». Ainsi, les regroupements des événements $V_2$, et $V_3$ se font au prix d'une *imprécision* sur la chronologie de leurs occurrences.

Les techniques proposées dans la littérature (nous décrivons quelques unes dans Section 2) ne permettent pas d'extraire ce type d'informations. Par exemple, pour les mêmes séquences et une contrainte de support équivalente, l'algorithme *GSP* présenté dans Srikant et Agrawal (1996) extrait la séquence suivante : $\langle (V_1)(V_2\ V_3)(M_1)\rangle$. Cette séquence représente la même série d'évènements fréquents que celle retournée par notre approche mais ne donne aucune information temporelle. Elle est interprétée comme suit : « Le vol $V_1$ est effectué, il est suivi par les vols $V_2$ et $V_3$ qui sont effectués *dans n'importe quel ordre* dans un intervalle de largeur 1, ces derniers sont eux-mêmes suivis par la tâche $M_1$ ». Vu le manque d'information temporelle concernant les enchainements des évènements, cette séquence est difficilement exploitable par un expert qui cherche à anticiper les applications de futures réparations, sachant que grâce à cette anticipation, d'importantes économies peuvent être réalisées en réduisant le temps d'immobilisation des avions.

La section suivante présente un état de l'art succinct sur l'ESF. Nous insistons surtout sur la différence de notre approche vis à vis de celles qui exploitent des intervalles temporels. Ensuite, nous donnons les définitions formelles et les propriétés des séquences que l'on cherche à extraire. La Section 4 décrit le processus d'extraction qui est une modification de l'algorithme *PrefixSpan* Pei et al. (2001). Nous terminons par une expérimentation permettant de comparer notre approche avec une technique de la littérature, en l'occurrence *GSPM* de Hirate et Yamana (2006), et une conclusion avec quelques perspectives.

# 2 État de l'art

Plusieurs travaux ont porté sur le regroupement d'évènements et l'extraction de séquences fréquentes par intervalles. Pham et al. (2009) permettent le regroupement de certains évènements en utilisant une notion de fenêtre glissante. Ce regroupement se fait lors d'une phase de pré-traitement des données sur lesquelles un algorithme d'extraction est par la suite exécuté. Cependant, les éléments regroupés sont associés à une estampille temporelle discrète : la plus petite des estampilles des évènements du groupe. Un choix arbitraire qui n'est nullement motivé, si ce n'est des considérations de simplification et qui de plus constitue une perte d'information. Ce faisant, le nombre de séquences obtenues est plus grand que le nombre de séquences initiales (une même séquence peut donner lieu à plusieurs regroupements différents). Se pose alors le problème de l'interprétation du support d'une séquence. Aussi, l'algorithme TCLW (pour Time Constraints LevelWise) présenté dans Masseglia et al. (2009) applique un regroupement d'évènements a travers l'application d'une fenêtre glissante et intègre des contraintes sur la distance qui sépare deux transactions successives. Cependant, les motifs extraits ne présentent pas d'estampillage temporel ce qui réduit les possibilités de leur exploitation.

Partant de données initialement estampillées par des intervalles, d'autres travaux ont été proposés afin d'en extraire les séquences fréquentes. Il est important de noter que toutes les approches citées ci-dessous considèrent l'intervalle estampille associé à un évènement comme représentant une durée tout au long de laquelle l'évènement a lieu. Ce qui les distingue de notre approche où l'on considère l'intervalle comme une *incertitude* sur le moment exact lors duquel l'évènement a lieu. Giannotti et al. (2006) extraient des séquences avec un algorithme à la *A Priori*. Ils identifient dans un premier temps les séquences fréquentes indépendamment des

estampilles. Par la suite, pour une séquence extraite, ils intersectent les intervalles temporels estampilles de ses occurrences pour obtenir, une nouvelle suite d'intervalles qui lui sont associés. Guyet et Quiniou (2011) représentent les estampilles temporelles d'une séquence sous forme d'hyper-cube dont les axes sont les évènements de la séquence. La similarité entre deux séquences est exprimée en fonction du volume de l'intersection de leurs hypercubes respectifs. Grâce à cette similarité, les auteurs regroupent les séquences. Un représentant d'un groupe (*i.e.* une séquence) est extrait si le groupe en question est *assez dense*. Wu et Chen (2007) et Chen et al. (2010) utilisent la théorie des intervalles d'Allen (1983) qui identifie treize relations entre deux intervalles. Un algorithme d'extraction à la *PrefixSpam* Pei et al. (2001) est appliqué sur des séquences à estampille temporelle par intervalles. Le résultat de ces deux approches consiste en un ensemble de relations *fréquentes* entre évènements et non pas des séquences avec des estampilles associées aux transactions. On note cependant le travail de Hirate et Yamana (2006) qui, comme l'approche que nous présentons, considère des séquences à estampilles temporelles discrètes en entrée et extrait des séquences fréquentes par intervalles. Les auteurs utilisent une fonction par paliers qui s'apparente à une fenêtre non glissante. Ainsi, des évènements très proches temporellement peuvent se retrouver dans des groupes différents du fait de l'application des paliers.

Par ailleurs, l'on distingue essentiellement deux techniques adoptées par les algorithmes d'ESF : La première s'apparente à *A priori* et a été utilisée dans, par exemple, Srikant et Agrawal (1996); Giannotti et al. (2006); Rabatel et al. (2009). La seconde utilise une stratégie *diviser pour régner* par réductions progressives de l'espace de recherche et sélection d'un ensemble de séquences 1-fréquentes pour la construction des séquences de longueur $k$ à partir d'une séquence de longueur $k-1$ (exemple, Hirate et Yamana (2006); Pei et al. (2001); Chen et al. (2010); Fournier-Viger et al. (2008); Wu et Chen (2007) et Guyet et Quiniou (2008)). Cette dernière s'avère moins coûteuse en parcours des données, donc en temps de calcul. C'est pour cette raison que l'algorithme que nous proposons s'inscrit dans cette deuxième catégorie. Il est inspiré de *PrefixSpan* utilisé dans Pei et al. (2001).

# 3 Notations et définitions

Dans un premier temps nous définissons les séquences temporelles à estampilles discrètes telles qu'elles sont énoncées dans plusieurs travaux traitant l'ESF à partir de séquences temporelles (Hirate et Yamana (2006); Fournier-Viger et al. (2008); Pei et al. (2001)).
Soit $\omega = \{e_1, e_2, \ldots, e_k\}$ un ensemble d'évènements. Une transaction est un ensemble d'évènements simultanés. Une séquence temporelle est une séquence de transactions ordonnées chronologiquement. A chaque transaction est associée une estampille temporelle. Une séquence temporelle $S$ notée $S = \langle (t_1, I_1), (t_2, I_2) \ldots (t_n, I_n) \rangle$ avec $n \in \mathbb{N}$ où $\forall 1 \leq i \leq n$ , $I_i$ est une transaction et $t_i$ est son estampille temporelle. Une base de séquences temporelles $D$ est une collection de séquences $S$ où chacune est identifiée par un identifiant unique *id_sequence*. Le support d'une séquence $S$ dans une base de séquences $D$ est noté $support_D(S)$. Il correspond au pourcentage de séquences de $D$ qui contiennent $S$. $S$ est fréquente dans $D$ si et seulement si son support est supérieur ou égal à un seuil minimal *minsupp* fixé par l'utilisateur.

Nous définissons maintenant les séquences temporelles par intervalles. Nous rappelons que ces séquences associent aux transactions des intervalles représentant une incertitude des

occurrences des évènements d'une transaction. Par exemple, soit $S = \langle([m, M], I)\rangle$, et que $I = \{e\}$ une transaction contenant le seul évènement $e$. Intuitivement, $S$ relate le fait suivant : « $e$ se produit à un moment *ponctuel* entre les instants $m$ et $M$ ».

**Définition 1 (Séquence temporelle par intervalles (STI))** *Soit $\omega = \{e_1, e_2, \ldots, e_k\}$ un ensemble d'évènements. Une séquence temporelle par intervalles (STI) $S$ de longueur $n$ est notée :$S = \langle([m_1, M_1], I_1), ([m_2, M_2], I_2) \ldots ([m_n, M_n], I_n)\rangle$ où $([m_i, M_i], I_i)$ est une transaction à estampille temporelle sous forme d'intervalle telle que :*

- *Les estampilles des transactions d'une séquence sont relatives à l'occurrence de son premier évènement : $m_i = m_i - m_1$ et $M_i = M_i - m_1$ ;*
- *$\forall 1 \leq i \leq n : m_i \leq temps\_occurrence(e_j) \leq M_i$. pour tout $e_j \in I_i$ ;*
- *Une séquence temporelle par intervalles est cohérente si pour deux transactions successives $I_i$ et $I_{i+1}$ on a : $m_i \leq m_{i+1}$ et $M_i \leq M_{i+1}$.*

**Exemple 1** *Soit $S1 = \langle([0, 1], A)([2, 2], BC)\rangle$ une STI, elle exprime le fait suivant : « A se produit avec une occurrence ponctuelle dans l'intervalle $[0, 1]$, et B et C se produisent simultanément au plus tôt une unité temporelle après l'évènement $A$ et au plus tard deux unités temporelles après ». $S2 = \langle([0, 3], A)([1, 2], B)([2, 5], C)\rangle$ n'est pas une STI cohérente, puisque la borne supérieure du second intervalle est inférieure à la borne supérieur du premier intervalle ($2 \lneq 3$).*

Concernant les séquences temporelles à estampilles discrètes, les transactions sont associées à un instant temporel ponctuel pendant lequel les évènements ont lieu. Il n'y a donc aucune incertitude sur le moment de leurs occurrence. On peut donc dire que chaque séquence temporelle à estampille discrète $S = \langle(t_1, I_1) \ldots (t_n, I_n)\rangle$ peut être transformée en une STI dont la largeur des intervalles est nulle (l'incertitude est nulle), elle est donc notée : $STI(S) = \langle([m_1, M_1], I_1), \ldots, ([m_n, M_n], I_n)\rangle$, avec $\forall 1 \leq i \leq n$, $m_i = M_i = t_i$.

Afin d'adapter les délais temporels des séquences aux besoins spécifiques d'extraction et de formulation, des contraintes temporelles sont considérées. Ces contraintes permettent de : fixer un seuil maximal d'incertitude, de réguler les espacements entres les transactions successives d'une séquence et d'en fixer une longueur minimale et maximale.

Soit $SI$ une STI de longueur $n$. SI satisfait les contraintes temporelles : $mingap$, $maxgap$, $min\_whole\_interval$, $max\_whole\_interval$ et la fenêtre glissante $ws$ si et seulement si $\forall\ 1 \leq i \leq n$ :

- *Gap* régule les distances temporelles minimale et maximale entre deux transactions successives :
$$mingap \leq (m_i - M_{i-1}) \leq maxgap$$
- *Whole_interval* régule la longueur minimale et maximale d'une séquence :
$$min\_whole\_interval \leq |m_1 - M_n| \leq max\_whole\_interval$$
- *Fenêtre glissante* permet de regrouper des évènements de transactions successives dans une même transaction en leur associant un intervalle temporel. Elle fixe sa largeur (incertitude) maximale :
$$|M_i - m_i| > ws$$

**Exemple 2** *Soit* $SI = \langle([0,1],A)([2,3],BC)([6,10],D)\rangle$ *une STI et les contraintes temporelles* $mingap$ *et* $maxgap$ *respectivement égales à 2 et 3. SI ne satisfait pas* $mingap$ *puisque* $m_2 - M_1 = 2 - 1 = 1 \leq 2$*. Par contre, SI satisfait* $maxgap$ *puisque pour toutes ses transactions successives* $maxgap$ *est satisfaite (*$m_2 - M_1 = 2-1 \leq 3$ *;*$m_3 - M_2 = 6-3 \leq 3$*). Pour une taille de fenêtre égale à 3, SI ne respecte pas la taille de la fenêtre puisque* $M_3 - m_3 = 10 - 6 \gneq 3$*. D'autre part, pour une fenêtre égale à 4, la contrainte est respectée par tous les intervalles de la séquence.*

Ces contraintes régulent l'effet du paramètre temporel dans une STI ; ils gèrent la distance minimale (respectivement maximale) entre deux transactions successives pour que la corrélation (i.e la mise en relation des co-occurrences) entre elles soit significative et que deux transactions trop proches ($mingap$) (respectivement trop éloignées ($maxgap$)) ne soient pas reliées. La taille de la fenêtre gère le regroupement des évènements et régit l'incertitude sur leurs occurrences. La contrainte $whole\_interval$, régule la longueur de la séquence afin que la corrélation de toutes les transactions de la séquence soit significative.

Dans ce qui suit nous définissons un opérateur $\diamondsuit$ pour regrouper des transactions successives d'une séquence. À partir d'une position $j$, $\diamondsuit$ joint des transactions qui se succèdent dans une STI $S$ en appliquant une fenêtre dont la taille fixe l'espacement temporel maximal entre la première et la dernière transaction du groupe. Le résultat est une séquence dont les transactions contiennent l'ensemble des évènements joints et ont pour estampille temporelle l'union des intervalles des transactions regroupées.

**Définition 2** *Soient une STI :* $SI = \langle([m_1,M_1],I_1)([m_2,M_2],I_2)\dots([m_n,M_n],I_n)\rangle$*, un entier* $j \leq n$ *et une taille de fenêtre* $ws$*. On définit l'opérateur* $\diamondsuit_{ws}$ *tel que :*

$$\diamondsuit_{ws}(SI,j) = SI' = \langle([m'_1,M'_1],I'_1)([m'_2,M'_2],I'_2)\dots([m'_n,M'_n],I'_k)\rangle$$

- *où* $\forall 1 \leq i \lneq j :\ ([m'_i,M'_i],I'_i) = ([m_i,M_i],I_i)$ *;*
- $\exists\ 1 \leqslant l_j \leqslant l_{j+1}, \dots l_i \dots \leqslant l_{k-1} \leqslant n$ *tel que :*
  - $I'_j = \cup_{p=j}^{l_j} I_p;\ \dots\ I'_i = \cup_{p=l_{i-1}+1}^{l_i} I_p;\ \dots\ I'_k = \cup_{p=l_{k-1}+1}^{l_n} I_p$,
  - $m'_j = m_j, M'_j = M_{l_j} \dots m'_i = m_{l_{i-1}+1}, M'_i = M_{l_i} \dots m'_k = m_{l_{k-1}+1}, M'_k = M_n$
  - $|m_j - M_{l_j}| \leqslant ws;\ \ \dots |m_{l_{i-1}+1} - M_{l_i}| \leqslant ws;\ \ \dots |m_{l_{k-1}+1} - M_n| \leqslant ws.$

**Exemple 3** *Soit* $SI = \langle([0,2],A)([1,2],B)([3,5],C)([4,6],D)\rangle$ *et une taille de fenêtre* $ws = 3$*, alors* $\diamondsuit_3(SI,1) = \langle([0,2],AB)([3,6],CD)\rangle$*. Les évènements des deux premières (respectivement des deux dernières) transactions sont regroupés et les intervalles qui leurs sont respectivement associés sont fusionnés car les deux fusions sont autorisées par la fenêtre (*$(2-0) \leqslant 3$ *respectivement* $(6-3) \leqslant 3$*). Notons que* $\diamondsuit_3(SI,2) = SI$*. En effet, lorsque le regroupement commence à la deuxième position, les intervalles de la deuxième et de la troisième transactions ne peuvent être fusionnés puisque leur union est trop large par rapport à la taille de la fenêtre* $ws$*. Finalement,* $\diamondsuit_3(SI,3) = \langle([0,2],A)([1,2],B)([3,6],CD)\rangle$ *et* $\diamondsuit_3(SI,4) = SI$*.*

Nous définissons maintenant l'opérateur $\overbrace{\diamondsuit}^{ws}$ qui pour une STI de longueur $n$ et une taille de la fenêtre $ws$ fournit un ensemble de STI's. L'opérateur applique la fenêtre sur une séquence en la faisant glisser et envisage toutes les possibilités de *fusion* des transactions successives.

Intuitivement, $\overbrace{\diamondsuit}$ fait glisser la fenêtre de regroupement et fournit l'ensemble de toutes les séquences condensées (combinaison de regroupement des transactions proches) qui peuvent représenter la séquence de départ. C'est l'ensemble des résultats des applications de $\diamondsuit_j$ sur une $n$-séquence pour des valeurs de $j$ dans $[1, n-1]$.

**Définition 3** *Soit* $SI = \langle([m_1, M_1], I_1) \dots ([m_n, M_n], I_n)\rangle$ *et* $ws$ *une taille de la fenêtre. Alors* $\forall 1 \leq i \lneq n$ *et* $SI_i = \diamondsuit_{ws}(SI, i)$ *on définit* $\overbrace{\underset{ws}{\diamondsuit}}$ *comme suit :*

$$\overbrace{\underset{ws}{\diamondsuit}}(SI) = \{SI_1, SI_2, \dots, SI_{n-1}\}$$

**Exemple 4** *Soit la séquence* $SI = \langle([0,2], A)([1,2], B)([3,4], C)([4,6], D)\rangle$ *et une fenêtre* $ws = 3$. *Alors* $\overbrace{\underset{3}{\diamondsuit}}(SI) = \{\langle([0,2], AB)([3,6], CD)\rangle, \langle([0,2], A)([1,4], BC)([4,6]D)\rangle, \langle([0,2], A)([1,2], B)([3,6]CD)\rangle\}$. *La séquence <([0, 2], AB)) ([3, 4], C) ([4, 6], D)> n'est pas incluse dans le résultat, par contre* $\langle([0,2], AB)([3,6], CD)\rangle$ *est « induite »de la précédente.*

À ce stade, nous pouvons définir la relation d'inclusion entre deux séquences temporelles par intervalles. Intuitivement une séquence temporelle par intervalles $SI$ contient une autre séquence par intervalles $SI'$, ssi les évènements de chaque transaction de $SI'$ sont contenus dans une (ou plusieurs) transaction(s) de $SI$ et que l'intervalle de la transaction de $SI'$ implique celui de la transaction de $SI$. L'ordre des transactions doit cependant être respecté.

**Définition 4** *Soient* $SI = \langle([m_1, M_1], I_1), \dots ([m_n, M_n], I_n)\rangle$, $SI' = \langle([m'_1, M'_1], I'_1) \dots ([m'_k, M'_k], I'_k)\rangle$ *et une taille de la fenêtre* $ws$. $SI'$ *contient* $SI$, *notée* $SI' \sqsupseteq SI$ *ssi :*

- $\exists\ 1 \leqslant l_{d1} \leqslant l_{f1} \leqslant l_{d2} \leqslant l_{f2}, \dots, \leqslant l_{dk} \leqslant l_{fk} = n$ ;
- $\exists\ 1 \leqslant i1 \leqslant i2 \dots \leqslant in \leqslant k$ ;
- $I'_{i1} \supseteq \sqcup_{p=l_{d1}}^{l_{f1}} I_p, \dots I'_{iu} \supseteq \sqcup_{p=l_{du}}^{l_{fu}} I_p, \dots I'_{in} \supseteq \sqcup_{p=l_{dn}}^{n} I_p$ ;
- $[0, (M_{f1} - m_{d1})] \supseteq [0, (M'_{i1} - m'_{i1})],\ \dots [(m_{du} - M_{f\supseteq(u-1)}), (M_{fu} - m_{d(u-1)})] \supseteq [(m'_{iu} - M'_{i(u-1)}), (M'_{iu} - m'_{i(u-1)})],\ \dots [(m_{dn} - M_{f(n-1)}), (M_{fn} - m_{f(n-1)})] \supseteq [(m'_{in} - M'_{f(n-1)}), (M'_{in} - m'_{d(n-1)})]$.

**Exemple 5** *Soient les* $SI_1 = \langle([0,2]A)([3,4], B)([5,6]C)\rangle$, $SI_2 = \langle([0,4]AB)\rangle$ *et* $SI_3 = \langle([0,2]A)([3,6]BC)\rangle$. $SI_1 \sqsupseteq SI_2$ *puisque* $[0,4]$ *implique* $[0,2]$ *et* $[3,4]$ *et* $SI_1 \sqsupseteq SI_3$. *Cependant* $SI_1 \not\sqsupseteq SI_4 = \langle([0,3]A)([2,6]BC)\rangle$ *puisque* $[0,2] \not\supseteq [0,3]$.

**Propriété 1** *Soient* $ws$ *une taille de fenêtre,* $S$ *une séquence temporelle à estampilles discrètes et* $SI$ *une STI.* $S$ *contient* $SI$ *si et seulement s'il existe* $SI' \in \overbrace{\underset{ws}{\diamond}}(STI(S))$ *et* $SI' \supseteq SI$. $\subseteq$ *est l'inclusion classique entre séquences (e.g. Agrawal et Srikant (1995)).*

Après avoir défini les STI et l'opérateur d'inclusion, nous pouvons redéfinir la notion de support d'une séquence temporelle par intervalles dans une base de séquences temporelles à estampilles discrètes. Nous considérons que le support d'une STI dans une base de séquences est le nombre de séquences de la base qui la contiennent au moins une fois.

**Définition 5** *Le support d'une STI* $SI$ *dans* $D$ *est défini par :* $supp_D(SI) = |\{S \in D \mid S \sqsupseteq SI\}|$. *Pour des besoins de simplification,* $supp_D(SI)$ *sera noté* $supp(SI)$.

# 4 Extraction de STI

Cette section décrit le procédé d'extraction de séquences temporelles par intervalles fréquentes à partir de séquences temporelles à estampilles discrètes. L'algorithme que nous proposons est appelé *STI-PS* (pour séquences temporelles par intervalles- *PrefixSpan*), il utilise une fenêtre glissante afin de regrouper progressivement des évènements proches (en respectant la taille de la fenêtre) et les ramener à une même transaction. Un tel regroupement permet d'unifier des occurrences d'évènements distincts et leur associe un intervalle temporel qui balaye l'ensemble des instants d'occurrences initiaux de chacun.

Notre algorithme applique une approche *pattern growth* (Pei et al. (2001)) qui utilise une méthode de recherche verticale de séquences basée sur la projection de la base. Dans un premier temps, l'ensemble des 1-séquences (évènements fréquents) est extrait. Il est noté $L_1 = \{S; S = \langle([m = 0, M = 0], e)\rangle; supp(e) \geqslant minsupp\}$. Ensuite, l'extraction continue récursivement par construction des $k+1$-séquences à partir d'une $k$-séquence. A chaque itération $i$, on applique deux étapes :

- une nouvelle projection de la base de séquences $D'$ est associée à chaque 1-STI de $L_1$ ($L_1$ étant identifiée dans l'espace de recherche associé à la récursion $i-1$). $D'$ représente ainsi l'espace qui contient les éventuelles continuations de la séquence construite à l'itération $i-1$.
- La seconde étape consiste à identifier l'ensemble des 1-STI à partir de $D'$ et à associer à chaque évènement fréquent l'intervalle qui représente son estampille temporelle. Chaque 1-STI identifiée est concaténée à la séquence extraite à l'itération précédente $(i-1)$ pour construire une nouvelle $i$-séquence. Dès lors, une nouvelle itération est exécutée.

Le procédé est arrêté si l'une des deux conditions suivantes est satisfaite : (1) Lorsque la projection est vide (étape 1) ou (2) lorsqu'aucune 1-STI n'est identifiée (étape 2).

Tout comme *GSPM* (Hirate et Yamana (2006)), *STI-PS* applique le principe de projection présenté ci-dessus. Cependant, il s'en distingue par la considération de la fenêtre glissante à deux niveaux : (1) la sélection d'évènements fréquents et (2) la projection de l'espace de recherche. Pour la sélection d'évènements fréquents, l'application de la fenêtre glissante intervient en associant des occurrences *décalées* d'un même évènement. Ces occurrences doivent appartenir à une même fenêtre. Cette modification permet de calculer la fréquence d'un évènement en appliquant la relaxation temporelle contrôlée par la taille de la fenêtre.

**Exemple 6** *Pour la séquence $SI = \langle([0,0]A)\rangle$, un support égal à 2, $ws = 2$ et la projection d'un espace de recherche $\mathcal{D}$ sur $SI$ $\mathcal{D}|_{SI} = \{\langle(1,B)(2,CD)\rangle, \langle(2,D)(3,B)(4,F)\rangle\}$, deux évènements fréquents sont relevés :* $([1,3]B)([2,2]D)$. *En effet la séquence* $([1,3]B)$ *permet de dire que dans l'espace de recherche l'évènement $B$ est fréquent et que ses occurrences varient sur 2 unités temporelles* $(3-1)$ *par rapport aux occurrences de $A$.*

Concernant la projection des espaces de recherche par un évènement fréquent, l'application de la fenêtre permet de prendre en considération l'aspect glissant. Nous définissons la projection de façon à ce qu'elle prenne en considération une exploration en arrière des évènements. Cette exploration est restreinte à la taille de la fenêtre. Une telle projection considère le désordre local (à la taille de la fenêtre) des évènements proches. Afin d'éviter l'extraction multiple d'une même séquence de longueur $k$ à partir d'un motif $SI$ de longueur $(k-1)$ le retour rétroactif ne considère pas les évènements qui ont déjà été traités pour extraire des fréquents de longueur $k$ à partir de $SI$. Nous définissons ainsi la projection d'une séquence comme suit :

**Définition 6** *Soient un ensemble d'évènements triés* $\omega = \{e_1, e_2 \dots e_m\}$*, une séquence temporelle* $S = \langle (t_1, I_1) \dots (t_n, I_n) \rangle$ *et une 1-STI* $([m, M], e_r)$*. Supposons qu'il existe* $j$ $(1 \leq j \leq n)$ *tel que* $e_r \in I_j$ *et* $t_j \in [m, M]$*. Nous avons alors :*

- *Le préfixe* $S$ *par rapport à* $([m, M], e_r)$ *est la sous-séquence de* $S$ *qui précède* $e$ *dans* $S$*. Il est noté :*

$$wprefixe(S, e_r, t_j) = \langle (t_1, I_1), (t_2, I_2) \dots (t_j, I_j) \rangle$$

- *Le suffixe de* $S$ *par rapport à* $([m, M], e_r)$ *est la sous-séquence de* $S$ *qui peut être considérée comme la continuation de* $([m, M], e_r)$

$$wsuffixe(S, e_r, [m, M]) = \langle (t_k, I'_k) \dots (t_p, I'_p \setminus \{e_r\}) \dots (t_u, I'_u) \dots (t_n, I_n) \rangle$$

*telle que :*

1. $t_p \in [m, M]$ *;*
2. $e_r \in I_p$ *;*
3. $t_k \leqslant (t_i - ws)$ *et* $t_{k-1} \leqslant (t_i - ws)$ *;*
4. $t_u \leqslant (t_i + ws)$ *et* $t_{u+1} \leqslant (t_i + ws)$ *;*
5. $I'_l = I_l \setminus \{e_1, e_2, \dots, e_{(r-1)}\}$.

*Soit* $\mathcal{D}$ *une base de séquences temporelles et* $\alpha = \langle ([m, M], I) \rangle$ *une 1-séquence fréquente. La projection de* $\mathcal{D}$ *par* $SI$ *est définie par :*

$$\mathcal{D}|_\alpha = \{SI \mid SI = wsuffix(S, I, [m, M]), S \in \mathcal{D}\} \; et \; ([m, M], i) \sqsubseteq S$$

**Exemple 7** *Dans l'exemple précédent, si on considère l'évènement fréquent* $([1, 3]B)$*, la séquence extraite devient* $SI = \langle ([0, 0]A)([1, 3]B) \rangle$*. La projection définie par notre approche permet de considérer tous les éléments autour du dernier évènement de* $SI$*. Elle fournit l'espace de recherche suivant* $\mathcal{D}|_{SI} = \{\langle (1, CD) \rangle, \langle (-1, D)(1, F) \rangle\}$*. Dans cet espace, l'évènement* $([-1, 1]D)$ *est fréquent puisque* $D$ *apparait dans les deux séquences et ses estampilles sont proches relativement à la taille de la fenêtre :* $1 - (-1) = 2 \leqslant 2$*, ce qui permet de construire le motif* $SI' = \langle ([0, 0]A)([2, 2]D)([1, 3]B) \rangle$ *qui sera représenté par la séquence* $SI' = \langle ([0, 0]A)([1, 3]BD) \rangle$*. Pour l'extraction du motif* $\langle ([0, 0]A) \; ([2, 2]D) \rangle$ *l'évènement* $B$ *n'est pas pris en compte dans le retour arrière afin de ne pas extraire deux fois la séquence* $SI'$*.*

Cette section a présenté *STI_PS* un algorithme d'extraction de STI's fréquentes en utilisant la méthode *pattern growth*. L'algorithme extrait les $k + 1$-séquences à partir de $k$-séquences en réduisant à chaque itération la taille de la base. Les intervalles temporels sont instaurés par l'application d'une fenêtre glissante à deux niveaux de l'algorithme : l'identification d'évènements fréquents et la projection de la base.

# 5 Expérimentation

Cette section présente une comparaison des séquences extraites par l'algorithme STI-PS avec celles obtenues par *GSPM* l'algorithme présenté dans (Hirate et Yamana, 2006). Les deux algorithmes sont basés sur *prefixSpan*. Ils se distinguent par le fait qu'ils utilisent des méthodes

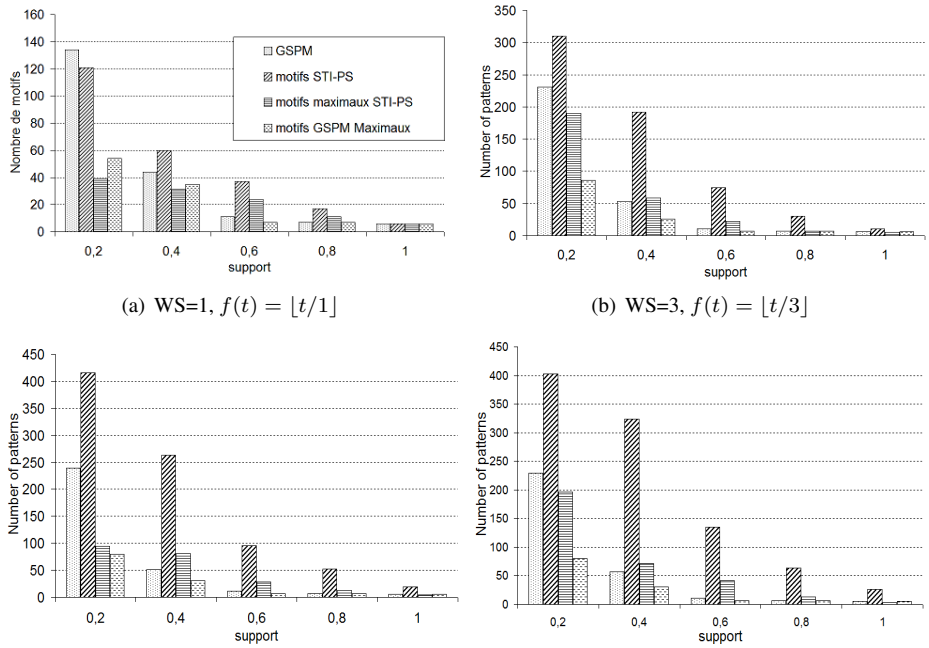
différentes pour regrouper les transactions. En effet, *GSPM* se base sur une fonction par pallier qui s'apparente à une fenêtre non glissante alors que notre algorithme utilise une fenêtre glissante. Pour que la comparaison ait un sens, à chaque fois qu'on fixe la valeur de la fenêtre glissante $ws$ pour notre algorithme, on fixe la fonction pallier de *GSPM* à $f(t) = \lfloor 1/ws \rfloor$. Nous donnons ci-dessous un exemple expliquant le fonctionnement de *GSPM* et renvoyons le lecteur à (Hirate et Yamana, 2006).

**Exemple 8** *Considérons la base de séquences* $\{S_1 = \langle (0, A)(1, B)(2, C)(3, F)(4, B)(6, G)\rangle$, $S_2 = \langle (0, A)(1, C)(2, B)(3, D)(4, F)(5, G)\rangle\}$*, un support minimal* $minsupp = 2$*, une fenêtre glissante* $ws = 2$ *et une fonction par paliers* $f(t) = \lfloor t/2 \rfloor$*. Les intervalles associés par* GSPM *seront donc de la forme* $[2 \times f(t), 2 \times (f(t) + 1)[$*. L'algorithme extrait d'abord les 1-séquences fréquentes suivantes A, B, C, F et G (l'estampille de toutes ces 1-séquences correspond à l'intervalle nul). En considérant la séquence B, la projection de la base fournit les séquences suivantes :* $\{S'_1 = \langle (1, C)(2, F)\ (3, B)(5, G)\rangle$, $S''_1 = \langle (2, G)\rangle$, $S'_2 = \langle (2, F)(3, G)\rangle\}$*. A partir de cet ensemble, le motif* $([2, 4[, F)$ *est identifié. Il est considéré comme fréquent car (1)* $F$ *apparaît dans* $S'_1$ *et* $S'_2$ *et (2) dans les deux cas,* $f(t) = \lfloor t/2 \rfloor = 1$*. Pour l'intervalle associé, on applique* $[2.f(t), 2.(f(t) + 1)[$ *ce qui donne* $[2, 4[$*. Dans la même projection,* $G$ *apparaît dans 3 séquences. Dans* $S''_1$ *et* $S'_2$ *on a* $f(t) = 1$ *alors que dans* $S'_1$ *on a* $f(t) = 2$*. Ainsi, seule* $([2, 4[, G)$ *est extraite.* $([4, 6[, G)$ *n'est pas considérée comme fréquente.*

Les deux algorithmes sont développés en JAVA en utilisant une version [1] de prefixSpan présentée dans (Fournier-Viger et al., 2008). Ils sont implémentés sur une machine Windows 7(64), Intel(R) Core(TM) 3 CPU 2.40 GHz Avec 3 GO RAM.

Nous comparons les séquences extraites par les deux méthodes en utilisant des données synthétiques. Les séquences contiennent 7 évènements différents et se caractérisent par un écart moyen entre les transactions de 3 unités temporelles et une longueur moyenne de 15 transactions par séquences. Lors de l'extraction, les contraintes mingap (respectivement maxgap, min_whole_interval et max_whole_interval) sont fixées à 0 (resp. 1, 0 et 15). La base de séquences utilisée contient 12 séquences. Nous avons délibérément choisi des jeux de données de petite taille car nous fondons notre étude non pas sur les temps d'exécution mais plutôt sur les résultats obtenus et plus précisément sur les nombres de séquences extraites. Dans cet article, notre objectif est de valider notre approche en vérifiant si les résultats extraits sont intéressants dans le cadre de l'application que nous ciblons. En effet, vu que notre approche est plus tolérante vis à vis de la chronologie des évènements, il est naturel de s'attendre à ce qu'on extraie *plus* d'informations. La Figure 1 illustre les résultats obtenus en faisant varier les différents paramètres. Pour chaque combinaison des valeurs des paramètres, nous avons mesuré le nombre de séquences extraites par les deux méthodes. De plus, pour chaque algorithme, nous avons calculé les séquences *maximales* à partir des résultats obtenus. Les figures 1(a) (resp. 1(b) 1(c) 1(d)) illustrent les variations des tailles des résultats sous différentes valeurs de regroupement. Elles montrent que le nombre de séquences extraites par *STI_PS* est beaucoup plus important que ceux obtenus par *GSPM*. Ceci est expliqué par l'application de contraintes temporelles plus souples. En effet, la fenêtre glissante regroupe les transactions de proche en proche et permet d'avoir toutes les combinaisons possibles de groupement mais aussi d'extraire des séquences plus longues. Aussi, le retour en arrière lors de la projection permet de prendre en compte plus d'évènements et certains y voient leur support augmenter. Le tableau 1 illustre

1. http ://www.philippe-fournier-viger.com/spmf/index.php




(a) WS=1, $f(t) = \lfloor t/1 \rfloor$ (b) WS=3, $f(t) = \lfloor t/3 \rfloor$

(c) WS=5, f(t)= $\lfloor t/5 \rfloor$ (d) WS=7, f(t)= $\lfloor t/7 \rfloor$

FIG. 1 – *Comparaison du nombre de séquences extraites en faisant varier le support, la taille de ws et la largeur du palier de la fonction*

le détail des nombres de séquences extraites par les deux méthodes pour une valeur de regroupement variable et un support égal à 0.4. Notons que lorsque les deux méthodes fournissent des séquences de même longueurs (correspondance de la figure 1(a) dans le tableau 1), les séquences maximales extraites par notre approche sont moins nombreuses que celles obtenues par *GSPM*. Ce cas de figure est illustré dans l'exemple 9. Cependant lorsque les séquences de *STI_PS* sont plus longues que celles retournées par *GSPM* les maximales sont plus nombreuses et leur nombre est majoritairement représenté par des motifs de longueurs supérieures aux séquences maximales extraites par *GSPM*. Notons enfin que le nombre de séquences maximales extraites par notre approche reste comparable à celles de *GSPM*.

**Exemple 9** *Si l'on reprend l'exemple 8, on peut vérifier que les plus longues séquences maximales que* GSPM *extrait sont :* $\langle([0,0[,B)([2,4[,F)\rangle$, $\langle([0,0[,G)\rangle$, $\langle([0,0[,A)\rangle$, $\langle([0,0[,C)\rangle$. *Celle extraite par* STI_PS *est* $\langle([0,2],ABC)([3,4],F)([5,6],G)\rangle$. *Si on considère la séquence* $\langle([0,0[,B)([2,4[,F)\rangle$ *extraite par* GSPM*, elle traduit le fait que* F apparaît dans l'intervalle $[2,4[$ après $B$. *La séquence* $\langle([0,2],ABC)([3,4],F)([5,6],G)\rangle$ *obtenue par* STI_PS *exprime, entre autres, le fait que* $F$ apparaît dans l'intervalle $[3-2=1, 4-0=4]$ après $B$. *Vu que* $[1,4]$ *contient* $[2,4[$*, on peut donc dire que la séquence maximale extraite par notre approche inclut toutes les séquences maximales extraites par* GSPM *et ce en tolérant plus d'incertitude.*

| ws | GSPM maximaux | STI-PrefixSpan | STI-PS maximaux |
|---|---|---|---|
| 1 | $L_1 = 13$, $L_2 = 21$, $L_3 = 1$ | $L_1 = 17$, $L_2 = 39$, $L_3 = 14$ | $L_2 = 21$, $L_4 = 14$ |
| 3 | $L_1 = 9$, $L_2 = 12$, $L_3 = 5$ | $L_1 = 7$, $L_2 = 53$, $L_3 = 96$, $L_4 = 26$, $L_5 = 3$ | $L_3 = 30$, $L_4 = 26$, $L_5 = 3$ |
| 5 | $L_1 = 9$, $L_2 = 13$, $L_3 = 2$ | $L_1 = 7$, $L_2 = 55$, $L_3 = 133$, $L_4 = 75$, $L_5 = 9$, $L_6 = 1$ | $L_3 = 26$, $L_4 = 42$, $L_5 = 13$, $L_6 = 1$ |
| 7 | $L_1 = 9$, $L_2 = 19$, $L_3 = 3$ | $L_1 = 7$, $L_2 = 51$, $L_3 = 115$, $L_4 = 88$, $L_5 = 21$, $L_6 = 4$ | $L_3 = 24$, $L_4 = 29$, $L_5 = 12$, $L_6 = 4$ |

TAB. 1 – *Nombre de i-séquences ($L_i$) extraites en fonction de la variation de la taille de $ws$ et un support égal à 0.4*

# 6 Conclusion

Cet article présente *STI-PS* un algorithme d'extraction de séquences en appliquant une fenêtre glissante qui permet une relaxation des contraintes temporelles. La fenêtre regroupe des évènements de transactions voisines de proche en proche permettant de considérer plusieurs combinaisons des données lors de l'extraction. L'algorithme est appliqué à une base de séquences à estampilles discrètes et en extrait des séquences fréquentes à estampilles temporelles par intervalles. L'estampille par intervalle représente une incertitude temporelle sur le moment d'occurrence des évènements de la transaction. L'ampleur de cette incertitude est régulée par la taille de la fenêtre glissante. Comparée à la solution proposée dans Hirate et Yamana (2006), notre approche permet d'extraire un plus grand nombre de séquences. De plus, certaines des séquences que nous obtenons sont plus longues que celles extraites par *GSPM*. Les futurs travaux porteront dans un premier temps sur l'optimisation de l'extraction des motifs maximaux. En effet, il n'est pas efficace de d'abord extraire *tous* les motifs et ensuite ne retenir que les maximaux. Dans un deuxième temps, nous comptons valider notre approche dans le cadre d'une application réelle. Il s'agit de pronostiquer l'occurrence de pannes dans des systèmes complexes à partir de données d'utilisation par le biais des séquences que nous obtenons.

# Références

Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of VLDB conference*.

Agrawal, R. et R. Srikant (1995). Mining sequential patterns. In *Proceeding of ICDE conference*. IEEE Computer Society Press.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of ACM 26*, 832–843.

Chen, Y.-C., J.-C. Jiang, W.-C. Peng, et S.-Y. Lee (2010). An efficient algorithm for mining time interval-based patterns in large database. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, Proceedings of CIKM conference, pp. 49–58.

Fournier-Viger, P., R. Nkambou, et E. M. Nguifo (2008). A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Proceeding of the 7th Mexican International Conference on Artificial Intelligence,*, pp. 765–778.

Giannotti, F., M. Nanni, D. Pedreschi, et F. Pinelli (2006). Mining sequences with temporal annotations. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pp. 593–597. ACM.

Guyet, T. et R. Quiniou (2008). Mining temporal patterns with quantitative intervals. In *Proceedings of The 4th International Workshop on Mining Complex Data*. IEEE Computer Society.

Guyet, T. et R. Quiniou (2011). Extracting temporal patterns from interval-based sequences. In *Proceedings of IJCAI conference*, pp. 1306–1311.

Hirate, Y. et H. Yamana (2006). Generalized sequential pattern mining with item intervals. *JCP. Journal of Computers 1*(3), 51–60.

Masseglia, F., P. Poncelet, et M. Teisseire (2009). Efficient mining of sequential patterns with time constraints: Reducing the combinations. *Expert Syst. Appl. 36*(2), 2677–2690.

Pei, J., J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, et M. Hsu (2001). Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of ICDE conference*, pp. 215–224.

Pham, Q.-K., G. Raschia, N. Mouaddib, R. Saint-Paul, et B. Benatallah (2009). Time sequence summarization to scale up chronology-dependent applications. In *Proceedings of CIKM conference*, pp. 1137–1146.

Rabatel, J., R. Bringay, et P. Poncelet (2009). So mad: Sensor mining for anomaly detection in railway data. In *Proceedings of Industrial Conference on Advances in Data Mining*, pp. 191–205.

Srikant, R. et R. Agrawal (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of EDBT conference*, pp. 3–17.

Wu, S.-Y. et Y.-L. Chen (2007). Mining nonambiguous temporal patterns for interval-based events. *IEEE Trans. on Knowl. and Data Eng. 19*, 742–758.

# Summary

Sequential patterns mining has been used in several domains. We note that time granularity is more or less important regarding the application domain. In this paper we propose a frequent *interval time sequences* (ITS) extraction technique from discrete temporal sequences using a sliding window approach in order to relax time constraints. The extracted sequences offer an interesting overview of the original data by allowing a temporal leeway on the extraction process. We formalize the ITS extraction under classical time and support constraints and conduct some experiments on synthetic data for validating our proposal.