Des Motifs Séquentiels Généralisés aux Contraintes de Temps Etendues

Céline Fiot, Anne Laurent, Maguelonne Teisseire

Des motifs séquentiels généralisés aux contraintes de temps étendues

Céline Fiot, Anne Laurent,
Maguelonne Teisseire

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
161 rue Ada
34392 Montpellier Cedex 5
{fiot, laurent, teisseire}@lirmm.fr

Résumé. Dans de nombreux domaines, la recherche de connaissances temporelles est très appréciée. Des techniques ont été proposées aussi bien en fouille de données qu'en apprentissage, afin d'extraire et de gérer de telles connaissances, en les associant également à la spécification de contraintes temporelles (e.g.: fenêtre temporelle maximale), notamment dans le contexte de la recherche de motifs séquentiels. Cependant, ces contraintes sont souvent trop rigides ou nécessitent une bonne connaissance du domaine pour ne pas extraire des informations erronées. C'est pourquoi nous proposons une approche basée sur la construction de graphes de séquences afin de prendre en compte des contraintes de temps plus souples. Ces contraintes sont relâchées par rapport aux contraintes de temps précédemment proposées. Elles permettent donc d'extraire plus de motifs pertinents. Afin de guider l'analyse des motifs obtenus, nous proposons également un niveau de précision des contraintes temporelles pour les motifs extraits.

1 Introduction

Dans un certain nombre de domaines (détection de fraudes, de défaillances, analyse de comportements), la recherche de connaissances temporelles est non seulement utile mais nécessaire. Certaines techniques d'apprentissage permettent de gérer et de raisonner sur de telles connaissances, (Allen, 1990) a notamment défini des opérations sur des règles associées à des intervalles de temps. Des techniques d'extraction de connaissances cherchent quant à elles à extraire des épisodes récurrents à partir d'une longue séquence (Mannila et al., 1997), (Raissi et al., 2005) ou de bases de séquences (Agrawal et Srikant, 1995), (Masseglia et al., 1998). La recherche de telles informations devient d'autant plus intéressante qu'elle permet de prendre en compte un certain nombre de contraintes entre les évènements comme par exemple la durée minimale ou maximale séparant deux évènements.

C'est dans ce cadre qu'a été introduite la recherche de motifs séquentiels généralisés dans (Srikant et Agrawal, 1996). Cette technique de fouille de données permet d'obtenir des séquences fréquentes respectant des contraintes spécifiées par l'utilisateur, à partir d'une base de données de séquences (par exemple les achats successifs de différents clients d'un supermarché). Différents algorithmes ont été proposés afin de gérer ces contraintes soit directement dans

le processus d'extraction, (GSP, Srikant et Agrawal (1996)) soit à l'aide d'un pré-traitement sur les séquences proposé dans GTC (Graph for Time Constraint), (Masseglia et al. (1999)).

Toutefois, si ces méthodes sont efficaces et robustes, elles ont pour principal inconvénient d'être spécifiées par l'utilisateur et nécessitent donc une bonne connaissance a priori des données et des durées à spécifier sous peine d'obtenir des connaissances peu pertinentes. Des travaux ont été proposés afin de déterminer de manière automatique la fenêtre optimale d'observation pour la recherche d'épisodes dans une séquence (Meger et Rigotti, 2004), mais ils sont difficilement adaptables à l'extraction de motifs séquentiels et dans ce domaine, aucun travail à notre connaissance ne propose une détermination automatique des contraintes de temps optimales. Par ailleurs, pour certaines applications il pourrait également être intéressant d'assouplir les contraintes spécifiées par les experts du domaine afin de compléter leurs connaissances. Enfin, le nombre de motifs séquentiels extraits, selon les contraintes de temps utilisés, peut rapidement devenir trop important pour que leur analyse soit efficace. Une mesure permettant l'exploitation des motifs séquentiels généralisés serait donc d'une grande utilité.

C'est pourquoi nous proposons d'étendre les contraintes de temps proposées pour l'extraction de motifs séquentiels généralisés en utilisant certains principes de la théorie des sous-ensembles flous. Notre méthode permet, en effet, à partir de contraintes de temps initiales et d'un degré de respect de ces valeurs, d'extraire des motifs séquentiels respectant des contraintes étendues et de fournir pour chacun d'eux sa précision temporelle. Nous offrons ainsi à l'utilisateur une flexibilité dans la spécification de ses contraintes ainsi qu'un outil d'analyse des nombreuses séquences fréquentes extraites.

Après avoir présenté les concepts fondamentaux associés aux motifs séquentiels et aux motifs séquentiels généralisés dans la section 2, nous présentons dans la section 3 notre définition des contraintes de temps étendues. La section 4 présente notre proposition d'algorithme mettant en oeuvre la gestion des contraintes de temps étendues lors du prétraitement des données. La section 5 présente ensuite quelques expérimentations montrant la faisabilité et l'efficacité de notre approche. Enfin, la section 6 conclut sur les perspectives qu'ouvrent nos travaux.

2 Des motifs séquentiels aux motifs séquentiels généralisés

2.1 Motifs séquentiels

Les motifs séquentiels ont initialement été proposés par Agrawal et Srikant (1995) et reposent sur la notion de *séquence fréquente maximale*.

Prenons par exemple une base de données DB d'achats pour un ensemble $\mathcal{C}$ de clients c. Une *transaction* t est un triplet <id_client, id_date, itemset> qui caractérise le client qui a réalisé l'achat, la date d'achat et les $items$ achetés.
Soit $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ l'ensemble des items de la base. Un *itemset* est un ensemble non vide et non ordonné d'items, noté $(i_1, i_2, \cdots, i_k)$. Une *séquence* se définit alors comme une liste ordonnée non vide d'itemsets s_i qui sera notée $< s_1 s_2 \cdots s_p >$. Une n-séquence est une séquence de taille n, c'est-à-dire composée de n items.

Exemple 1. *Si un client achète les produits a, b, c, d et e selon la séquence <(a)(b c)(d)(e)>, cela signifie qu'il a d'abord acheté le produit a, puis les produits b et c ensemble, ensuite seulement le produit d et finalement le produit e.*

Une séquence $S' =< s'_1 s'_2 \cdots s'_m >$ est une *sous-séquence* de $S =< s_1 s_2 \cdots s_p >$ s'il existe des entiers $a_1 < a_2 < \cdots < a_m$ tels que $s'_1 \subseteq s_{a_1}, s'_2 \subseteq s_{a_2}, \cdots, s'_m \subseteq s_{a_m}$, S' est *incluse* dans S.

Exemple 2. *La séquence* $S' =< (b)(e) >$ *est une sous-séquence de* S *car* $(b) \subseteq (b\ c)$ *et* $(e) \subseteq (e)$. *Par contre* $< (b)(c) >$ *n'est pas une sous-séquence de* $< (b\ c) >$, *ni l'inverse.*

Les transactions de la base sont regroupées par client et ordonnées chronologiquement, définissant ainsi des séquences de données. Un client c *supporte* une séquence S si elle est incluse dans la séquence de données du client c. Le *support* d'une séquence est alors défini comme le pourcentage de clients de la base DB qui supporte S. Une séquence est dite *fréquente* si son support est au moins égal à une valeur minimale $minSupp$ spécifiée par l'utilisateur.

La recherche de motifs séquentiels dans une base de séquences telle que DB consiste alors à trouver toutes les séquences maximales (non incluses dans d'autres) dont le support est supérieur à $minSupp$. Chacune de ces séquences fréquentes maximales est un motif séquentiel.

2.2 Motifs séquentiels généralisés

Telle qu'elle a été introduite ci-dessus, la notion de séquence présente une certaine rigidité pour de nombreuses applications. En effet, si l'intervalle de temps entre deux transactions successives est très court, on pourrait envisager de les considérer comme simultanées. A l'inverse, deux évènements trop éloignés peuvent ne pas avoir de lien entre eux. C'est pourquoi la notion de séquence généralisée a été proposée par Srikant et Agrawal (1996) afin de pallier ces restrictions en introduisant la prise en compte de contraintes temporelles.

Ces contraintes sont au nombre de trois : $mingap$ est une durée minimale que l'on doit respecter entre deux itemsets successifs d'une même séquence ; $maxgap$ est l'écart maximal dans lequel doivent se trouver deux itemsets successifs ; $windowSize$ est la fenêtre dans laquelle les items de deux transactions différentes peuvent être regroupés dans un même itemset. On modifie alors la notion d'inclusion décrite précédemment pour tenir compte de ces contraintes.

Une séquence de données $d =< d_1 \cdots d_m >$ supporte une séquence $s =< s_1 \cdots s_n >$ s'il existe des entiers $l_1 \leq u_1 < l_2 \leq u_2 < \cdots < l_n \leq u_n$, donnant les dates de début et de fin des itemsets $d_1 \cdots d_n$, tels que $s_i \subset \cup_{k=l_i}^{u_i} d_k$, $1 \leq i \leq n$ et

- date(d_{u_i}) - date(d_{l_i}) $\leq windowSize, 1 \leq i \leq n$; (1)
- date(d_{l_i}) - date($d_{u_{i-1}}$) $> mingap, 2 \leq i \leq n$; (2)
- date(d_{u_i}) - date($d_{l_{i-1}}$) $\leq maxgap, 2 \leq i \leq n$; (3)

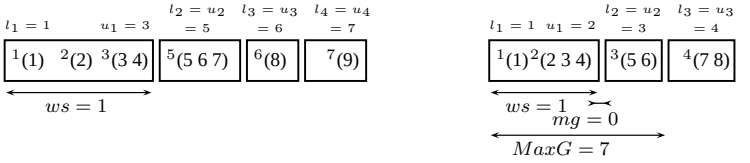
Exemple 3. *Soit les contraintes* $windowSize$=1, $mingap$=0 *et* $maxgap$=7, *les séquences d1 et d2 présentées* TAB. *1 et la séquence candidate* s=*<(1 2 3 4)>.* $d1$ *et* $d2$ *supportent-elles* s *?*

	Date	1	2	3	4	5	6	7
d1	**Client 1**	1	2	3 4	-	5 6 7	8	9
d2	**Client 2**	1	2 3 4	5 6	7 8	-	-	-

TAB. 1 – *base exemple*

s *apparaît en regroupant les trois premiers itemsets de la séquence du client 1 (*FIG. *1-(a)). On a alors* $date(d_{l_1})$=1 *et* $date(d_{u_1})$=3, *on ne respecte pas la contrainte (1). Donc* s *n'est pas*

*incluse dans la séquence $d1$ du client 1. Pour le client 2 (*FIG. *1-(b)), en regroupant les dates 1 et 2, $date(d_{l_1})$=1 et $date(d_{u_1})$=2, on respecte les trois contraintes, s est incluse dans $d2$.*



(a) séquence du client 1 (b) séquence du client 2

FIG. 1 – *Représentation des contraintes de temps $windowSize$ (ws), $minGap$ (mg) et $maxGap$ ($MaxG$) sur les séquences des clients 1 et 2.*

3 Vers des contraintes de temps étendues

Dans cette section, nous allons examiner la mise en œuvre des contraintes de temps étendues par analogie avec la théorie des sous-ensembles flous pour ensuite proposer leur définition et celle de la précision temporelle des séquences soumises à de telles contraintes.

3.1 Mise en œuvre

La théorie des sous-ensembles flous, introduite par Zadeh (1965) autorise l'appartenance partielle à une classe et donc la gradualité de passage d'une situation à une autre. Cette théorie constitue une généralisation de la théorie ensembliste classique, des situations intermédiaires entre le tout et le rien étant admises. Dans ce cadre, un objet peut donc appartenir partiellement à un ensemble et en même temps à son complément.

On considère par exemple l'univers des tailles possibles d'un individu. Un sous-ensemble flou A (*Petit* ou *Grand* par exemple) est défini par une fonction d'appartenance μ_A qui décrit le degré avec lequel chaque élément de l'univers considéré appartient à A, ce degré étant compris entre 0 et 1. Ainsi, un individu de 1m63 pourra à la fois être grand et petit avec, par exemple, un degré de 0.7 pour le sous-ensemble flou *Grand* et 0.3 pour le sous-ensemble flou *Petit*.

Les opérateurs en logique floue sont une généralisation des opérateurs classiques. On considère notamment la négation, l'intersection et l'union. L'opérateur $\top$ ou *t-norme* (norme triangulaire) est l'opérateur binaire d'intersection : $\mu_{A\cap B}(x) = \top(\mu_A(x), \mu_B(x))$. L'opérateur $\perp$, *t-conorme*, (conorme triangulaire) est l'opérateur d'union : $\mu_{A\cup B}(x) = \perp(\mu_A(x), \mu_B(x))$. Nous noterons $\overline{\top}$ (resp. $\underline{\perp}$) l'opérateur $\top$ (resp. $\perp$) généralisé au cas n-aire.

Notre proposition d'extension des contraintes de temps pour les motifs séquentiels est fondée sur une analogie avec la théorie des sous-ensembles flous. Ainsi, on ne souhaite plus simplement qu'une séquence respecte des contraintes spécifiées mais permettre à l'utilisateur de relaxer ces contraintes jusqu'à un certain seuil. Ce seuil correspond à un degré minimal de satisfaction des contraintes temporelles. Il sera spécifié par l'utilisateur pour chacune des contraintes, de même que leur valeur initiale. Soit ws, g et G ces valeurs initiales pour les

contraintes $windowSize$, $minGap$ et $maxGap$. On considère ρ_{ws}, ρ_{mg} et ρ_{MG} les niveaux de précision associés à chacune d'elles, ces niveaux étant compris entre 0 et 1, 0 indique alors que l'on souhaite parcourir l'ensemble des valeurs possibles et 1 que le paramètre correspondant est fixe. L'utilisation de tels degrés implique de considérer plusieurs valeurs pour les paramètres $windowSize$, $mingap$ et $maxgap$. Il est alors possible de considérer plusieurs "chemins" ou séquences dans les achats des clients pour reconstruire un motif. Cependant chacune de ces séquences respecte "plus ou moins" la contrainte initiale ce qui est évalué par les niveaux de précision ρ_{ws}, ρ_{mg} et ρ_{MG} dont le calcul est détaillé au paragraphe 3.3.

3.2 Extension des contraintes de temps

La définition des contraintes de temps étendues est basée sur les valeurs limites utiles que les différents paramètres peuvent prendre. Ces valeurs utiles correspondent aux valeurs limites au-delà desquelles on ne pourra générer de séquences candidates respectant les contraintes.

Prenons l'exemple de $windowSize$. Dans le cas classique, $windowSize$ prend une valeur fixe ws et la condition (2.2) signifie que $date(d_{u_i}) - date(d_{l_i}) \in [\![0, a]\!], 1 \leq i \leq n$. Reprenons l'ensemble des clients $\mathcal{C}$. Pour chaque client c, ses transactions ont des dates comprises entre $D_{c_{min}}$ et $D_{c_{max}}$. La contrainte (2.2) impose alors, en considérant que $date(d_{u_i}) \leq D_{c_{max}}$ et $date(d_{l_i}) \geq D_{c_{min}}$, que $windowSize \leq M = \max_{c \in \mathcal{C}}(D_{c_{max}} - D_{c_{min}})$: au plus, $windowSize$ vaut l'écart maximum pour tous les clients entre les dates minimales et maximales des transactions. L'idée est donc de faire varier la valeur de $windowSize$ entre sa valeur spécifiée ws et sa valeur maximum utile, M. Deux cas peuvent alors se présenter : soit $a \geq M$, dans ce cas l'algorithme classique suffit, soit $a < M$ et dans ce cas, il faut compléter les parcours effectués dans le cas classique. Afin de générer d'autres séquences candidates, on va donc faire varier (ici linéairement) la valeur de $windowSize$ entre ws et M, comme le montre la FIG. 2. Pour une valeur x de $windowSize$, la précision des séquences générées sera donnée par une fonction d'appartenance $\mu_{ws}(x) = \frac{1}{ws-M}x - \frac{M}{ws-M}$, pour $x \geq a$ et 1 sinon. La valeur maximale ws_ρ de $windowSize$, pour un seuil de précision ρ_{ws} est donc : $ws_\rho = \lfloor (ws - M)\rho_{ws} + M \rfloor$.

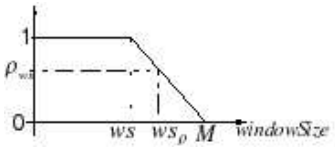


FIG. 2 – *Variation de la précision selon la valeur de* $windowSize$

De même, on a pour $maxgap$: $\mu_{MG}(x) = \frac{1}{G-M}x - \frac{M}{G-M}$ et $G_\rho = \lfloor (G - M)\rho_{MG} + M \rfloor$, ainsi que $\mu_{mg}(x) = \frac{1}{g-m}x - \frac{m}{g-m}$ et $g_{\rho_{mg}} = \lceil (g - m)\rho_{mg} + m \rceil$ pour $mingap$. Le détail des calculs pour ces deux contraintes sont présentés dans Fiot et al. (2005a).

Exemple 4. *Reprenons l'exemple 3. On cherche à savoir si la séquence s=<(1 2 3 4)> est incluse dans les séquences des clients C1 et C2, en utilisant cette fois les paramètres* ws*=1, g=0 et G=7, avec* ρ_{ws}*=0.5,* ρ_{mg}*=1 et* ρ_{MG}*=1. Seul le paramètre* $windowSize$ *varie. Avec* M*=6 et* ws*=1, on obtient* a_ρ*=2 :* $windowSize$ *prend donc successivement les valeurs 1 et 2.*

Pour $windowSize$=1, on retombe sur l'exemple 3, s est incluse dans $d2$ mais pas dans $d1$. Pour $windowSize$=2, pour le client 1, on a $date(d_{l_1})$=1 et $date(d_{u_1})$=3 qui respecte bien la contrainte (2.2). Donc s est incluse dans $d1$ et toujours dans $d2$.

3.3 Précision temporelle d'une séquence

Pour chacune des séquences fréquentes trouvées à la fin du processus d'extraction, on calcule la précision avec laquelle chacune d'elles respecte les contraintes de temps. On définit la précision d'une séquence s pour un client c comme le degré de respect simultané des trois contraintes de temps (1), (2) et (3), calculé à l'aide d'une t-norme ($\top$). Pour chaque client, on cherche, parmi toutes les séquences d'achats ς_c, l'occurrence de s qui respecte au mieux les contraintes de temps, en utilisant une t-conorme ($\perp$).

La précision temporelle d'une séquence $s =< s_1 \cdots s_n >$ pour le client c est donnée par :

$$\varrho(s,c) = \underline{\perp}_{s \in \varsigma_c} \Big(\overline{\top}_{i \in [1,n]} \big(\mu_{ws}(date(s_{u_i}) - date(s_{l_i}))\big), \overline{\top}_{i \in [2,n]} \big(\mu_{mg}(date(s_{l_i}) - date(s_{u_{i-1}})), \mu_{MG}(date(s_{u_i}) - date(s_{l_{i-1}}))\big)\Big) \quad (4)$$

Pour l'ensemble de la base, le degré de respect des contraintes de temps est donné par la moyenne des degrés de chacun des clients, c'est-à-dire :

$$\Upsilon(s) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \varrho(s,c) \quad (5)$$

4 GETC - Graph for Extended Time Constraints

L'algorithme GTC proposé dans (Masseglia et al., 1999) permet de transformer une séquence d'un client en un graphe de séquences respectant les contraintes de temps. Les graphes de séquences des différents clients sont ensuite utilisés pour déterminer les séquences fréquentes par un algorithme d'extraction de motifs séquentiels tel que PSP Masseglia et al. (1998). L'efficacité de cette approche ayant été démontrée dans (Masseglia et al., 1999), nous avons choisi de nous en inspirer pour développer la nôtre. Nous proposons donc un algorithme permettant de construire un graphe de séquences pour les contraintes de temps étendues et également de calculer la précision des motifs séquentiels généralisés extraits.

4.1 GETC - les algorithmes

A partir d'une séquence d'entrée d, l'algorithme GETC construit son graphe de séquences G_d. Cet algorithme regroupe un certain nombre de sous-fonctions ($addEdge$, $propagate$, $pruneMarked$ et $convertEdges$), non détaillées ici, qui sont présentées de manière plus approfondies dans (Fiot et al., 2005a).

GETC commence par créer les sommets correspondant aux itemsets de la séquence puis ajoute à l'ensemble des sommets l'ensemble des combinaisons d'itemsets permises selon les différentes valeurs de $windowSize$. Pour cela, l'algorithme $addWindowSize$ (non présenté ici) parcourt chaque sommet x et détermine pour chacun d'entre eux quels autres sommets y peuvent être "fusionnés" avec x (si y.date() - x.date() $\leq ws$). Chaque sommet correspond

GETC - **Input :** d, une séquence de données
Ouput : $G_d(S, A)$, le graphe de séquences associé à d, S l'ensemble des sommets de G_d, A l'ensemble des arêtes

Pour chq itemset $i \in d$ **faire**
  S = S $\cup\{i\}$; newLevel() ;
Fin Pour
addWindowSize(S) ;
Tant que ($x \neq$ S.fi rst()) **faire**
  $l \leftarrow x$.level().prec() ; $mg \leftarrow g$;
  Tant que (x.begin() - l.end() $\leq mg$) **faire**
    $contmg \leftarrow$ FALSE ;
    Si (x.begin() $> l$.end()) **Alors**
      [sinon sommet inclus]
      Tant que ($mg \geq g_\rho$) **faire**
        Si (x.begin() - l.end() $> mg$) **Alors**
          $contmg \leftarrow$ TRUE ; $mg \leftarrow g_\rho$-1 ;
        Sinon
          mg - - ;
        Fin Si
      Fin Tant que
    Fin Si
    Si ($contmg$ == FALSE) **Alors**
      propagate(x,l) ; $l \leftarrow l$.prec() ;
    Fin Si
  Fin Tant que
Pour chq $w \in l$ **faire**
  $included \leftarrow$ TRUE ;
  $MG \leftarrow G$;
  Tant que ($MG \leq G\rho$) **faire**
    Si (x.end() - w.begin() $\leq MG$) **Alors**
      addEdge(w,x) ;
      $MG \leftarrow G_\rho + 1$;
    Sinon
      MG++ ;
    Fin Si
  Fin Tant que
Fin Pour
$x \leftarrow$ S.next(x) ;
Fin Tant que
pruneMarked($G_d(S, A)$) ;
convertEdges($G_d(S, A)$) ;
Retourner $G_d(S, A)$;

ALG. 1: GETC

alors à un itemset x.itemset() qui possède une date de début x.begin() et une date de fin x.end(). Ces sommets sont regroupés en *niveau* par date de fin des itemsets. Cela permet de tester le respect des contraintes pour un niveau et non pour chaque sommet. On accède à l'ensemble des prédécesseurs d'un sommet x par x.prev() et à ces successeurs par x.succ().

Ensuite pour chacun des sommets du graphe de séquences, GETC ajoute les arcs respectant les contraintes $minGap$ et $maxGap$. Ainsi, pour chaque sommet, on cherche le premier niveau accessible pour la contrainte $minGap$ (ie. l.begin() - x.end() $> g_\rho$) et pour chaque sommet z de ce niveau, on construit les arcs (x,z), pour chaque sommet z tel que z.end() - x.begin() $\leq maxGap$. La fonction $addEdge$ permet d'éviter les inclusions de chemins, grâce à la construction d'arcs temporaires dans des cas d'inclusions possibles.

L'algorithme $addEdge$ construit les arcs entre des sommets qui respectent les contraintes de temps sur $minGap$ et $maxGap$. Il crée un arc définitif si les sommets ne sont pas déjà liés par une séquence ou une inclusion de leurs successeurs ou prédécesseurs. Dans ce cas, l'arc construit est temporaire et ne devient définitif que si la séquence qu'il forme est maximale. C'est également lors de l'exécution de cet algorithme que les sommets inclus sont marqués, pour pouvoir ensuite être supprimés s'ils sont inutiles.

Dans le cas où pour un sommet x, on ne peut atteindre le niveau l à cause du non respect de la contrainte $mingap$, on utilise la fonction $propagate$ pour "propager ce saut". Pour chacun des sommets y de ce niveau inaccessible, on ajoute si nécessaire et si on respecte les contraintes $minGap$ et $maxGap$, un arc entre chacun des successeurs de x et ce sommet y. Comme dans $addEdge$, on construit des arcs temporaires ou définitifs selon que la séquence construite peut

éventuellement être incluse ou au contraire n'a aucune chance de l'être.

Enfin, l'algorithme $pruneMarked$ élimine les sommets de sous-séquences incluses. Puis, $convertEdges$ supprime les arcs de sous-séquences incluses : pour tout arc temporaire de x vers y, si y est inclus dans un successeur z de x et si les successeurs de y sont également tous des successeurs de z, alors il existe une sous-séquence incluse, l'arc est supprimé. Sinon, l'arc est indispensable pour obtenir toutes les séquences maximales.

GETC étant utilisé comme prétraitement pour la prise en compte de contraintes temporelles en vue de l'extraction de motifs séquentiels, il doit générer absolument toutes les séquences issues d'une séquence de données. Par ailleurs, afin d'améliorer le temps d'extraction, il est nécessaire que GETC n'extraie que les séquences maximales, nous avons donc montré dans (Fiot et al., 2005a) que l'algorithme GETC construit exactement toutes les séquences de la plus grande taille possible pour les séquences respectant $windowSize$, $minGap$ et $maxGap$.

4.2 Construction du graphe de séquences

Nous utilisons GETC comme prétraitement pour la prise en compte des contraintes de temps étendues et PSP (Masseglia et al., 1998) pour l'extraction des motifs séquentiels. Cette approche du type générer-élaguer utilise une structure d'arbre prefixé pour organiser les séquences candidates et permettre de trouver plus efficacement l'ensemble des candidats inclus dans une séquence de données. En utilisant ainsi le graphe de séquences obtenu par GETC, la vérification des contraintes de temps est rendue inutile pendant le parcours des candidats, seule l'inclusion devant être vérifiée. Cette méthode est similaire à celle proposée dans (Masseglia et al., 1999) qui permet d'optimiser l'extraction de motifs séquentiels généralisés grâce à un parcours pour les contraintes de temps indépendant et sans retour arrière de l'arbre des séquences candidates et ainsi la vérification du moins de combinaisons possibles.

Exemple 5. *On dispose de la base exemple* TAB. *2 et des paramètres suivants pour les contraintes de temps : pour* $windowSize$, ws*=2 et* $\rho_{ws} = 0.87$*, donc* a_ρ*=4 ; pour* $maxGap$, G*=4 et* $\rho_{MG} = 0.85$ *donc* G_ρ*=6 ; pour* $minGap$, g*=2 et* $\rho_{mg} = 0.5$*, donc* g_ρ*=1.*

Date	**1**	**3**	**4**	**5**	**6**	**8**	**9**	**10**	**12**	**17**	**18**
Client 1	1	-	2 3	3 4	4	4	-	5	6	7	8
Client 2	2 3	4	-	-	5	6	-	-	-	-	-
Client 3	1 2	-	3	3 4	4	-	5 6	-	-	-	-

TAB. 2 – *Base exemple*

Nous allons présenter ici la construction du graphe de séquences pour la séquence de données du client 1. La première étape consiste en la création de l'ensemble des sommets initiaux, correspondants aux itemsets des transactions présentés dans la base de données TAB. *2.*

La deuxième étape consiste à prendre en compte $windowSize$ *et les dates des différentes transactions afin de créer des sommets combinaisons, grâce à la fonction* $addWindowSize$*. La troisième étape est celle de la construction des arcs respectant les contraintes* $minGap$ *et* $maxGap$*, grâce à l'algorithme principal, ainsi que les fonctions* $propagate$ *et* $addEdge$*. On obtient le graphe* FIG. *3. La fonction* $pruneMarked$ *supprime ensuite les sommets marqués. Enfin, les arcs temporaires sont convertis en arcs définitifs ou supprimés par la fonction* $convertEdges$*. La* FIG. *4 présente le graphe de séquences final.*

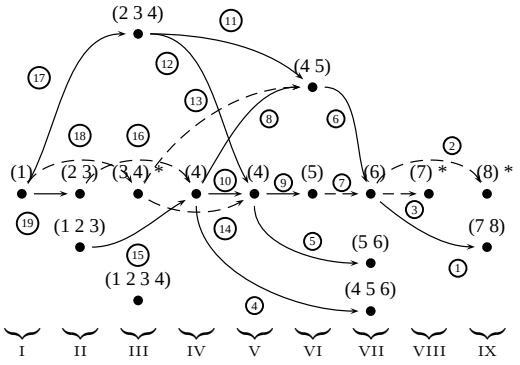


FIG. 3 – *Graphe de séquence après la deuxième étape (prise en compte de $mingap$ et $maxgap$ étendus)*

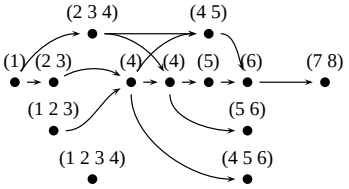


FIG. 4 – *Graphe de séquences final pour le client 1*

4.3 Calcul de la précision temporelle d'une séquence

Une fois le graphe de séquences construit, on connaît toutes les séquences autorisées par les contraintes de temps et celles qui sont interdites. Cependant, certaines séquences respectent les contraintes fortes de l'utilisateur alors que d'autres ont été construites en appliquant les contraintes étendues, elles ne sont donc pas équivalentes. On calcule alors la précision de chacun des chemins (séquences maximales) et on l'affecte aux sous-séquences qui le composent.

Afin de déterminer l'appartenance du chemin vis-à-vis des contraintes de temps, on value chaque arc (x,y) par $\top(\mu_{mg}(y.\text{begin}()\text{-}x.\text{end}()),\mu_{MG}(y.\text{end}()\text{-}x.\text{begin}()))$. Chaque sommet est valué par $windowSize$. La précision d'une sous-séquence, et donc le degré de respect des contraintes de temps est alors donnée par la formule (5). Grâce à l'algorithme $valueGraph$ (non présenté ici), le graphe est alors parcouru et à chaque sommet s, on attribue la valuation $\mu_{ws}(s.\text{end}()\text{-}s.\text{begin}())$ et à chaque arc entre s et t, la valuation $\top(\mu_{mg}(t.\text{begin}()\text{-}s.\text{end}()),\mu_{MG}(t.\text{end}()\text{-}s.\text{begin}()))$.

Exemple 6. *A partir de la base de données* TAB. *2 et des contraintes de temps spécifiées dans l'exemple 5, on construit les trois graphes de séquences correspondants. En prenant $minSupp$=70%, on obtient six séquences maximales fréquentes : <(2 3 4)>, <(2 3)(4)(5 6)>, <(2)(4 5)>, <(3 4)(5)>, <(3 4)(6)> et <(3)(4 5)>. Elles ont toutes un support de 1. Il reste à déterminer la précision de respect des contraintes de temps initiales. Pour cela, chacun des graphes de séquences est valué comme précisé dans la section précédente. Les sommets construits avec $windowSize$=0,1,2 ont un degré de 1, avec $windowSize$=3, un degré de 0.93 et avec $windowSize$=4, un degré de 0.87. De même les arcs construits avec $mingap$=1 ont un degré pour $mingap$ de 0.5 et de 1 pour $mingap$=2. En ce qui concerne $maxgap$, le degré est de 1 pour $maxgap \leq$4, de 0.92 pour $maxgap$=58 et 0.85 pour $maxgap$=6. La* FIG. *5 présente le graphe valué de la séquence du client 2. On utilise ensuite ces valuations pour calculer le degré de respect des contraintes de temps par les motifs extraits (*TAB. *3).*

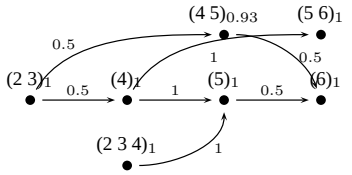


FIG. 5 – *Graphes de séquences valués pour le client 2*

Motifs séquentiels	ϱ_{CI1}	ϱ_{CI2}	ϱ_{CI3}	Υ
$< (2\,3\,4) >$	1	1	0.87	0.96
$< (2\,3)(4)(5\,6) >$	0.5	0.5	0.5	0.5
$< (2)(4\,5) >$	0.85	0.5	1	0.78
$< (3\,4)(5) >$	0.85	1	1	0.95
$< (3\,4)(6) >$	0.5	0.5	1	0.67
$< (3)(4\,5) >$	0.85	0.5	0.5	0.62

TAB. 3 – *Calcul de la précision pour les motifs séquentiels extraits*

5 Expérimentations

Ces expérimentations ont été réalisées sur un PC équipé d'un processeur 2,8GHz et de 2Go de mémoire DDR, sous systèmes Linux, noyau 2.6. Le but de ces mesures est de comparer le comportement de GETC par rapport à GTC. Pour certaines comparaisons, nous utilisons également une implémentation de PSP, intégrant ou non la gestion des contraintes de temps. Les résultats présentés ici ont été obtenus à partir du traitement de plusieurs jeux de données synthétiques comportant environ 1000 séquences de 20 transactions en moyenne. Chacune de ces transactions comportant en moyenne 15 items choisis parmi 1000.

La première étape a consisté à comparer les temps d'exécution en l'absence de contrainte de temps, c'est-à-dire en prenant $windowSize = 0$, $minGap = 0$ et $maxGap = \infty$ pour GTC ainsi que pour GETC, avec une précision de 1 pour les trois paramètres. Ainsi, nous avons pu comparer le temps d'exécution de notre algorithme avec ceux de PSP et GTC. La figure FIG. 6(a) montre que le comportement de GETC est similaire à celui de GTC et que les temps d'exécution sont quasiment identiques, pour des motifs extraits qui sont les mêmes.

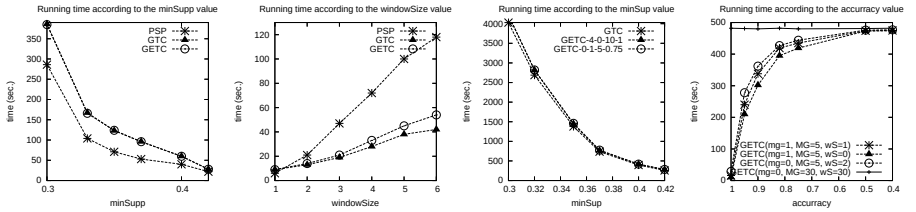
Nous avons ensuite répété ces mesures en introduisant le traitement de contraintes de temps, toujours avec une précision de 1 afin de comparer le comportement de GETC et GTC pour la gestion de contrainte de temps non-étendues. La FIG. 6(b) montre l'évolution du temps d'extraction en fonction de la valeur de $windowSize$. GETC a un comportement linéaire proche de celui de GTC. La différence provient de la phase de traitement de la précision, dont le temps augmente légèrement avec $windowSize$, puisque si ce paramètre augmente, on augmente le nombre de sommets dans le graphe de séquences.

La deuxième partie de nos expérimentations a porté sur l'analyse des motifs séquentiels extraits par GETC en fonction de la précision des différentes contraintes. La figure FIG. 6(c) présente les temps d'extraction comparés de GTC et GETC en fonction du support minimum selon des valeurs choisies des différents paramètres. Ces valeurs ont été calculées afin que les contraintes de temps utilisées pour GTC et GETC avec une précision de 1 correspondent aux valeurs limites de GETC avec une précision différentes de 1. Les paramètres retenus sont :

- GETC avec $windowSize$=0, $minGap$=1 et $maxGap$=5 avec une précision de 0.75, qui nous donne $ws_\rho = 4$, $mg_\rho = 0$ et $MG_\rho = 10$,
- GETC avec $windowSize$=4, $minGap$=0 et $maxGap$=10 avec une précision de 1,
- GTC avec $windowSize$=4, $minGap$=0 et $maxGap$=10,

On peut constater que l'utilisation de GETC avec des contraintes de temps étendues n'est pas plus coûteuse que celle de GTC avec les valeurs limites des contraintes, tout en permettant d'obtenir les mêmes motifs séquentiels, accompagnés de leur précision temporel. Il est inté-

ressant, dans le cas où on ignore la valeur optimale d'une ou plusieurs contraintes de temps, d'utiliser GETC avec une précision différente de 1 pour certains paramètres, afin de balayer un ensemble de possibilités plus large. L'analyse des motifs obtenus et de leur précision pourra renseigner sur une valeur plus adéquate des contraintes de temps.
Enfin, la FIG. 6(d) montre l'évolution du temps d'extraction en fonction de la précision pour un support minimum de 0.37. On constate que le temps d'extraction atteint une valeur limite qui correspond à la valeur maximale utile des trois paramètres de contraintes de temps.



(a) (b) (c) (d)

FIG. 6 – *Temps d'extraction : (a) en fonction de minSupp en ne considérant aucune contrainte temps (i.e. $windowSize$=0, $minGap$=0, $maxGap$=∞ and $\rho_{ws} = \rho_{MG} = \rho_{mg} = 1$) ; (b) en fonction de windowSize avec $minGap$=2, $maxGap$=∞ and $minSup$=0.35 (pour GETC, $\rho_{ws} = \rho_{MG} = \rho_{mg} = 1$) ; (c) en fonction de minSupp en considérant des contraintes de temps avec une précision de 1 ou non ; (d) en fonction de la précision selon différentes valeurs des contraintes de temps ($minSupp$=0.37).*

6 Conclusion et perspectives

Les motifs séquentiels généralisés présentés par Srikant et Agrawal (1996) permettent une définition plus large de l'inclusion en introduisant l'utilisation de contraintes de temps. Toutefois, cette définition reste encore trop rigide, notamment dans le cas où l'utilisateur n'a qu'une vague idée des contraintes temporelles qui lient ses données. Dans cet article nous proposons donc une extension des contraintes de temps pour les motifs séquentiels généralisés, qui permet plus de souplesse dans la spécification des paramètres de contraintes temporelles. Notre approche se base sur la construction de graphes de séquences pour intégrer les contraintes de temps dans le processus d'extraction de motifs séquentiels. La faisabilité et la robustesse de cette méthode ont été montrées pour GTC dans Masseglia et al. (1999). Le principe de GETC étant le même, nous avons pu montrer son efficacité pour résoudre le probléme de la recherche de séquences généralisées avec des contraintes de temps, étendues ou non, et la similitude de son comportement avec celui de GTC. Nous avons également pu mettre en évidence la flexibilité offerte par la mise en place des contraintes étendues, il nous reste encore à en valider la robustesse. Enfin, nous envisageons d'étendre les motifs séquentiels flous présentés dans Fiot et al. (2005b) à des motifs séquentiels généralisés, avec contraintes de temps étendues ou non.

Références

Agrawal, R. et R. Srikant (1995). Mining Sequential Patterns. In *11th International Conference on Data Engineering*, Taipei, Taiwan, pp. 3–14. IEEE Computer Society Press.

Allen, J. F. (1990). Maintaining knowledge about temporal intervals. *Readings in qualitative reasoning about physical systems*, 361–372.

Fiot, C., A. Laurent, et M. Teisseire (2005a). Contraintes de temps étendues pour les motifs séquentiels. Technical Report 5056, LIRMM.

Fiot, C., A. Laurent, et M. Teisseire (2005b). Motifs séquentiels flous : un peu, beaucoup, passionnément. In *5èmes journées d'Extraction et Gestion des Connaissances (EGC '05)*, pp. 507–519.

Mannila, H., H. Toivonen, et A. I. Verkamo (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery 1*(3), 259–289.

Masseglia, F., F. Cathala, et P. Poncelet (1998). The PSP Approach for Mining Sequential Patterns. In *Principles of Data Mining and Knowledge Discovery*, pp. 176–184.

Masseglia, F., P. Poncelet, et M. Teisseire (1999). Extraction efficace de motifs séquentiels généralisés : le prétraitement des données. In *15 ème Journée Bases de Données Avancées (BDA '99)*, pp. 341–360.

Meger, N. et C. Rigotti (2004). Constraint-based mining of episode rules and optimal window sizes. In *8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pp. 313–324. Springer-Verlag.

Raissi, C., P. Poncelet, et M. Teisseire (2005). Need for speed : Mining sequential patterns in data streams. In *21 ème Journée Bases de Données Avancées (BDA '05)*.

Srikant, R. et R. Agrawal (1996). Mining sequential patterns : Generalizations and performance improvements. In *5th International Conference on Extending Database Technology (EDBT '96)*, London, UK, pp. 3–17. Springer-Verlag.

Zadeh, L. (1965). Fuzzy sets. *Information and Control 3*(8), 338–353.

Summary

Mining temporal knowledge can be really useful in lots of fields. Some methods have been proposed as well in data mining as in machine learning in order to extract and manage such knowledge using temporal constraints. In particular some works have been proposed to mine generalized sequential patterns. However these constraints are often too crisp or need a very precise assessment to avoid erroneous information. Within this context we propose an approach based on graph of sequence built from extended temporal constraints. These relaxed constraints enable us to find more generalized sequential patterns. We also propose to measure the temporal accuracy of extracted sequences compared to the initial constraints.