

S²MP : une mesure de similarité pour les motifs séquentiels

Hassan Saneifar, Sandra Bringay, Anne Laurent, Maguelonne Teisseire

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)
161 rue Ada 34392 Montpellier Cedex 5
Université montpellier 2
{saneifar, bringay, laurent, teisseire}@lirmm.fr

Résumé. Dans le domaine de l'extraction de connaissances, comparer la similarité des objets est une tâche essentielle, par exemple pour identifier des régularités ou pour construire des classes d'objets homogènes. Ce problème est très important pour les données séquentielles présentes dans divers domaines d'application (e.g. séries d'achats de clients, navigations d'internautes). Il existe des mesures de similarité comme Edit distance et LCS adaptées aux séquences simples. Cependant elles ne sont pas pertinentes dans le cas des séquences complexes composées de séries d'ensembles, comme les motifs séquentiels. Dans cet article, nous proposons une nouvelle mesure de similarité (S²MP - Similarity Measure for Sequential Patterns) prenant en compte les caractéristiques des motifs séquentiels. S²MP est une mesure paramétrable en fonction de l'importance accordée à chaque caractéristique des motifs séquentiels selon le contexte d'application, ce qui n'est pas le cas des mesures existantes. La qualité sémantique de notre mesure ainsi que son efficacité a été validée grâce à des expérimentations sur différents jeux de données. Les expérimentations montrent que les clusters obtenus en utilisant S²MP sont plus homogènes, plus précis et plus complets que ceux obtenus avec Edit distance.

1 Introduction

Dans de nombreux domaines d'applications comme la bio-informatique ou l'analyse des achats des clients, les données sont stockées sous forme séquentielle. On distingue différents types de **séquences** : (1) **les séquences d'items** composés d'éléments atomiques comme des séquences d'entiers et (2) **les séquences d'itemsets** constituées d'objets plus complexes eux-mêmes composés de plusieurs items.

On distingue également deux types de **séquences d'itemsets** : (1) **les séquences de données**, (2) **les motifs séquentiels fréquents**. Par exemple, dans une base d'achats de supermarché, une séquence de données correspond à la série de paniers d'un client (*les itemsets*) ordonnés chronologiquement. Chaque panier est composé des articles achetés (*les items*).

Les motifs séquentiels présentés par Agrawal et Srikant (1995) sont des schémas fréquents extraits à partir des séquences de données. Un motif séquentiel met en évidence des associations inter-transactions. Par exemple, un motif séquentiel extrait d'une base d'achats de supermarché met en évidence des comportements fréquents dans les achats des clients.

Par exemple, $\langle (Chocolat, Soda)(gâteaux, chips)(biscuit diététique) \rangle$ se traduit par : souvent les clients achètent du chocolat et du soda, puis au cours d'une prochaine visite au magasin, ils achètent des gâteaux et des chips et ensuite plus tard ils achètent des biscuits diététiques".

Les séquences de données et les motifs séquentiels partagent deux caractéristiques : (1) *chaque itemset est un ensemble non ordonné d'items*, (2) *les itemsets sont ordonnés dans la séquence*.

Dans plusieurs domaines d'applications, des motifs séquentiels sont extraits à partir des données afin d'identifier des corrélations. L'évaluation des motifs séquentiels extraits permet de mesurer la proximité de ces motifs et par conséquent d'identifier la cohérence ou incohérence dans les comportements découverts par les motifs extraits. Or, l'évaluation des motifs séquentiels extraits est une tâche difficile à cause des caractéristiques particulières des motifs séquentiels.

Pour de nombreuses applications, il est nécessaire d'évaluer la proximité des motifs séquentiels. Par exemple, pour réaliser un clustering de motifs séquentiels afin d'identifier des comportements similaires, pour extraire des motifs séquentiels sous contraintes ou pour la visualisation des motifs séquentiels. Ces comparaisons se basent sur une mesure de similarité qui est l'un des concepts centraux de la fouille de données (Moen, 2000). Pour une comparaison pertinente, la mesure de similarité doit être adaptée aux caractéristiques de données. Elle doit également passer à l'échelle pour supporter les gros volumes de données.

Pour une mesure de similarité dédiée aux motifs séquentiels, de nombreuses applications sont envisageables comme le clustering des motifs séquentiels pour identifier des comportements similaires, l'extraction de motifs séquentiels sous contraintes de similarité ou la visualisation de motifs séquentiels. Dans l'objectif de clarifier l'intérêt d'une mesure dédiée aux motifs séquentiels, nous détaillons certaines de ces applications où la comparaison de motif séquentiel est essentielle.

Une telle mesure peut être utilisée pour le regroupement (clustering) des motifs séquentiels en plusieurs groupes (clusters) correspondant à un type homogène de corrélations. Ces groupes sont utilisés pour créer des profils de comportements. Par exemple, dans le contexte de la détection d'anomalies, le clustering de motifs séquentiels extraits à partir des logs de connexion normaux peut être utilisé afin de créer des profils de comportements généraux correspondant à une représentation plus abstraite de ces comportements. Cette modélisation de comportements passe à l'échelle (Sequeira et Zaki, 2002). En outre, le clustering nous permet de détecter des déviations éventuelles dans les données. Dans une base d'achats de supermarché, le regroupement des motifs séquentiels peut aider à la segmentation de la clientèle ou à la prédiction en fonction des comportements d'achat.

L'extraction de motifs séquentiels sous contraintes de similarité est une autre application de la mesure de similarité. Des motifs séquentiels extraits par des techniques a priori (Agrawal et Srikant, 1995) sont très volumineux. Des techniques ont été développées pour extraire des motifs séquentiels jugés intéressants car respectant des contraintes comme des contraintes de similarité. Par exemple, Capelle et al. (2002) extrait uniquement des motifs étant similaires à un motif de référence.

L'interrogation d'un ensemble de motifs séquentiels est également un contexte où la mesure de similarité est utile. Étant donné un motif séquentiel considéré comme une requête, nous cherchons des motifs similaires. Ces interrogations sont très pertinentes en bio-informatique et plus généralement dans la visualisation des motifs séquentiels.

La définition de la similarité peut varier selon le type de similitudes que l'on cherche à identifier entre deux objets. Des mesures de similarité différentes peuvent refléter les différents visages des données et leur contexte. Deux objets peuvent être considérés comme très similaires par une mesure et très différents par une autre mesure (Moen, 2000). En outre, nous affirmons que la similitude ne doit pas être toujours symétrique, comme Tversky (1977) l'a souligné. Par exemple, nous disons "les Turcs se battent comme des Tigres" et non pas "les Tigres se battent comme des Turcs". Dans certaines applications telles que l'extraction de motifs séquentiels sous contraintes de similarité ou l'interrogation d'une base de motifs séquentiels, on compare des motifs à un motif de référence. Il s'agit d'une comparaison directionnelle où une mesure de similarité non-symétrique est applicable.

Plusieurs approches ont été développées pour comparer la similarité entre deux séquences notamment dans le domaine de la bio-informatique. Il existe un grand nombre de travaux portant sur les séquences d'items mais peu de travaux portent sur les séquences d'itemsets. Les mesures dédiées aux séquences d'items ne sont pas adaptées aux séquences d'itemsets surtout aux caractéristiques particulières de motifs séquentiels.

Afin de comparer la similarité des motifs séquentiels, nous définissons ici une mesure de similarité (**S²MP** : **S**imilarity **M**eaure for **S**equential **P**atterns) qui prend en compte les caractéristiques et la sémantique des motifs séquentiels. Cette mesure compare deux motifs au niveau des itemsets et de leurs positions dans la séquence ainsi qu'au niveau de la ressemblance des items dans les itemsets. Notre mesure résulte de la combinaison de deux scores : (1) le score relatif à la similarité des itemsets selon les items qui les composent ; (2) le score relatif à la similarité des itemsets selon leur position et leur ordre dans les séquences.

Par ailleurs, S²MP s'adapte à différents contextes car il est possible de modifier ou d'adapter chaque score. De plus, S²MP est une mesure paramétrable en fonction de l'importance accordée à chaque caractéristique des motifs séquentiels (*l'ordre ou la similarité des itemsets*) selon le contexte d'application. Cela rend notre mesure flexible et sensible aux caractéristiques du domaine, ce qui n'est pas le cas des mesures existantes.

La section 2 présente les travaux existant sur les mesures de similarité pour les motifs séquentiels fréquents. Notre mesure de similarité (S²MP) est présentée dans la section 3. Les résultats obtenus lors des expérimentations sur la mesure de similarité sont détaillés dans la section 4.

2 Inadéquation des mesures avec les motifs séquentiels

Les deux mesures de similarité principalement utilisées pour les séquences d'itemsets sont : **Edit distance** et **LCS**. Dans cette section, nous listons les inconvénients de ces deux mesures pour les motifs séquentiels. Ensuite, nous citons une troisième approche qui a été appliquée aux données multidimensionnelles.

La mesure *Edit distance* a été utilisée dans Capelle et al. (2002) pour extraire des motifs séquentiels sous contraintes de similarité. Les auteurs définissent un motif séquentiel comme une liste ordonnée de symboles appartenant à Σ où Σ est un alphabet comprenant un ensemble fini de symboles.

Exemple 2.1. Soit $M_1 = \{(ab)(c)\}$ et $M_2 = \{(a)(c)\}$, deux motifs séquentiels. On associe aux itemsets des symboles : $X = (ab)$, $Y = (c)$ et $Z = (a)$

Les opérateurs d'Edit distance étant appliqués sur les éléments de la séquence (*i.e. les itemsets*), la distance est donc le coût de substitution de X par Z . Un itemset est ici réduit à un événement caractérisé par les valeurs de certains attributs (*les items*). Un motif séquentiel est donc considéré comme une séquence d'événements¹ ordonnés en fonction de leurs occurrences (Mannila et Ronkainen, 1997; Moen, 2000). Par conséquent, les itemsets (ab) et (a) sont traités comme deux symboles (*événements*) différents. Or, (ab) et (a) peuvent être deux comportements similaires.

ApproxMAP développé par Kum et al. (2003) est un algorithme pour fouiller les séquences consensus² qui se déroule en deux phases : (1) le clustering de séquences (2) la fouille de motifs consensus directement à partir de chaque cluster. Dans la phase 1, les auteurs ont utilisé Edit distance comme mesure de similarité mais en remplaçant le coût de l'opérateur substitution par la *différence ensembliste normalisée*. Bien que cette modification surmonte la limitation d'Edit distance argumentée précédemment, les auteurs ont noté que la différence ensembliste donne plus de poids aux éléments communs. Elle est donc appropriée si les points communs sont plus importants que les différences.

En outre, Moen (2000) discute l'influence des coûts des opérations d'Edit distance sur le degré de similarité dans le cas des séquences. Elle indique qu'il est plus naturel de donner plus de poids à l'insertion (*ou suppression*) des ensembles rares qu'aux ensembles fréquents. L'influence du coût des opérations et le type des opérations sont justifiés dans (Moen, 2000).

Par ailleurs, Edit distance ne s'adapte pas à différentes définitions de la similarité. Par exemple, lorsque l'on cherche à comparer des motifs séquentiels extraits à partir de données bio-informatiques (*transcriptomiques*), le contenu des itemsets (items) est plus important que l'ordre des itemsets. Or, Edit distance ne permet pas de prendre en compte cette caractéristique liée aux données. Pour une comparaison pertinente et sémantiquement correcte, il faut comparer deux motifs séquentiels au niveau des itemsets et de leurs positions dans les séquences ainsi qu'au niveau des items dans les itemsets.

La mesure *LCS* (Longest Common Subsequence) est utilisée pour la comparaison des séquences Sequeira et Zaki (2002). Cette mesure donne la longueur de la sous-séquence commune la plus longue à deux séquences. Nous donnons ci-dessous trois limitations.

Exemple 2.2. Soit trois motifs séquentiels :

$M_1 = \{(\mathbf{bc})(\mathbf{df})(e)\}$, $M_2 = \{(a\mathbf{bc})(mn)(\mathbf{de})(gh)(fg)\}$ et $M_3 = \{(\mathbf{bc})(\mathbf{df})\}$.

La sous-séquence commune la plus longue est $\{(bc)(d)\}$.

$LCS(M_1, M_2) = 2$, $LCS(M_1, M_3) = 2$.

Premièrement, *LCS* ne prend pas en compte la position des itemsets (*dans l'ordre*) dans les deux séquences. La sous-séquence $\{(bc)(d)\}$ est une sous-séquence avec des itemsets consécutifs dans $M_1 = \{(bc)(df)(e)\}$ et $M_3 = \{(bc)(df)\}$ et non consécutifs dans $M_2 = \{(abc)(mn)(de)(gh)(fg)\}$.

Deuxièmement, *LCS* ne considère pas la longueur de la partie non-commune entre les deux séquences. La partie non commune dans M_2 (*i.e.* $\{(abc)(mn)(de)(\mathbf{egh})(\mathbf{fg})\}$) est plus longue que celle de M_3 (*i.e.* $\{(e)(bc)(df)\}$). C'est pourquoi la normalisation de *LCS* par le nombre d'items dans les séquences est nécessaire.

Troisièmement, le nombre d'items non-communs dans les itemsets (*où les items communs apparaissent*) n'influence pas la valeur de *LCS*. L'itemset (bc) de la sous-séquence est inclus dans (abc) de M_2 mais il est égal à (bc) de M_1 et de M_3 . Ce problème n'est pas résolu par la normalisation.

Plantevit et al. (2007) proposent une comparaison de la similarité entre deux motifs séquentiels multidimensionnels $S_1 = \{b_1, b_2, \dots, b_k\}$ et $S_2 = \{b'_1, b'_2, \dots, b'_k\}$:
 $d(s_1, s_2) = Op(dist(b_j, b'_j))$ pour $j = 1 \dots k$ avec *dist* une mesure de distance et *Op* un opérateur d'agrégation. La comparaison se fait entre des blocs³ correspondants. On compare le bloc i de S_1 avec le bloc i de S_2 . Ce type de comparaison n'est pas pertinent lorsqu'il existe un décalage dans l'une des séquences.

Les mesures présentes dans littérature présentent certains inconvénients dans le cas des motifs séquentiels. Une mesure de similarité dédiée aux motifs séquentiels doit prendre en compte :

¹Edit distance est fréquemment utilisée pour les séquences d'événements Moen (2000)

²Une séquence consensus est partagée par de nombreuses séquences et couvre de nombreux motifs courts.

³Ici, Un bloc de données peut être vu comme un itemset

1. les motifs séquentiels sont des séquences ordonnées d'itemsets (*ensemble d'items*) et non d'items,
2. les **positions** (*distance dans l'ordre*) des itemsets lors du calcul de la similarité,
3. le nombre d'**items communs** et **non communs** au niveau de la séquence et au niveau des itemsets correspondants,
4. idéalement, une mesure de similarité doit être modulaire et paramétrisable à chaque niveau de comparaison afin de pouvoir l'adapter aux contextes différents.

C'est en prenant en compte ces quatre critères que nous avons proposé la mesure S²MP.

3 S²MP : Description

La mesure de similarité (S²MP : Similarity Measure for Sequential Patterns) résulte de l'agrégation de deux scores :

- le **score de mapping** qui mesure la ressemblance de deux motifs en fonction des liens qu'il est possible d'établir entre les itemsets,
- le **score d'ordre** qui mesure la ressemblance des deux séquences vis-à-vis de l'ordre et de la position des itemsets.

Le déroulement de l'algorithme se fait en deux phases qui correspondent au calcul de ces deux scores.

Dans la phase 1, nous mettons en correspondance les itemsets des deux séquences en fonction de leur contenu et calculons pour chaque lien un poids (*similarité entre deux itemsets*) selon le nombre d'items communs et non communs. Le score de mapping est alors calculé en considérant l'ensemble des poids.

À la phase 2, l'objectif est de donner un score en fonction de la ressemblance de deux séquences selon l'ordre et la position des itemsets. Parmi tous les liens établis à la phase 1, nous cherchons ceux qui respectent l'ordre des itemsets dans les deux séquences. Cela signifie que nous rejetons les mappings croisés (cf. figure 2). Nous mesurons également la ressemblance des itemsets mis en correspondance au niveau de leurs positions dans les deux séquences. Finalement, le score d'ordre est calculé en fonction du pourcentage de liens respectant l'ordre des itemsets et du score mesurant la ressemblance des itemsets au niveau de leur position dans les séquences.

Phase 1 – Calcul du score de mapping. En entrée de ce calcul, nous avons les deux séquences à comparer. Nous établissons des liens entre chaque itemset i de la séquence 1 $Seq_1(i)$ et l'itemset j le plus ressemblant dans la séquence 2 $Seq_2(j)$. Nous évaluons ces liens par des poids.

Définition 3.1. $poids(i, j)$ entre le $i^{ème}$ itemset de la Seq_1 et le $j^{ème}$ itemset de la Seq_2 :

$$poids(i, j) = \frac{|Seq_1(i) \cap Seq_2(j)|}{(|Seq_1(i)| + |Seq_2(j)|) / 2}$$

Nous formalisons le lien (*mapping*) entre deux itemsets ainsi :

$$Mapping(Seq_1(i), Seq_2(j)) \mid poids(i, j) = \max_{x \in [0, |Seq_2|]} \{poids(i, x)\} \wedge poids(i, j) \neq 0$$

$$\text{Si } poids(i, j) = poids(i, k) \Rightarrow ts(j) < ts(k)$$

Si plusieurs liens sont possibles avec des poids égaux, nous prenons le lien qui est associé à l'itemset ayant un *timeStamp*⁴ (ts) inférieur, c'est-à-dire l'itemset qui est situé avant les autres dans l'ordre de la séquence.

Conflit. Un itemset sélectionné de la 2^{ème} séquence pour être mis en correspondance avec un itemset de la séquence 1 peut avoir déjà été utilisé dans un autre lien. Nous appelons cette situation "conflit" (cf. figure 1). Pour résoudre ce problème, nous recherchons un nouveau candidat non déjà lié avec la fonction "Résolution de conflit". Pour les deux

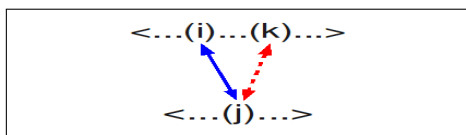


FIG. 1 – Conflit de mapping.

itemsets de séquence 1 en conflit ($Seq_1(i)$ et $Seq_1(k)$), nous cherchons quatre nouveaux candidats dans la Seq_2 (*autre que l'itemset candidat actuel* : $Seq_2(j)$) avant et après $Seq_2(j)$:

⁴Le *timeStamp* dans notre contexte, correspond à la position des itemsets dans l'ordre de la séquence. Par exemple dans la séquence (a)(ab)(c), le $ts(ab) = 2$.

- $nextMaxBefor_i$ et $nextMaxBefor_k$
- $nextMaxAfter_i$ et $nextMaxAfter_k$

Nous créons ensuite les quatre couples possibles entre ces candidats :

$$\begin{aligned} < Seq_1(i), Seq_2(j) >, < Seq_1(k), nextMaxBefor_k > & \quad < Seq_1(k), Seq_2(j) >, < Seq_1(i), nextMaxBefor_i > \\ < Seq_1(i), Seq_2(j) >, < Seq_1(k), nextMaxAfter_k > & \quad < Seq_1(k), Seq_2(j) >, < Seq_1(i), nextMaxAfter_i > \end{aligned}$$

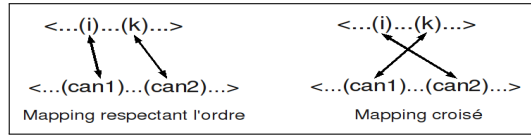


FIG. 2 – Mapping croisé et l'ordre des itemsets.

Nous considérons ici deux types de mises en relation : celles respectant l'ordre et les autres (cf. figure 2). Nous appelons ces dernières des mappings croisés : Supposons qu'un itemset à la position i de la première séquence soit mis en relation avec un itemset à la position i' dans la deuxième séquence. Un itemset à la position j (où $i < j$) est mis en relation avec un itemset à la position j' de la deuxième séquence. Ce lien est conforme à l'ordre si $i' < j'$ et non conforme (mapping croisé) si $i' > j'$.

Nous calculons la pertinence des quatre couples identifiés précédemment avec un score dit *coupleScore* dont le calcul dépend du type de mise en relation :

Si l'ordre est conforme :

$$coupleScore(i, Can_1)(k, Can_2) = \frac{Poids(i, Can_1) + Poids(k, Can_2)}{2}$$

Si l'ordre est non conforme (*mappings croisés*) :

$$coupleScore(k, Can_1)(i, Can_2) = \frac{1}{2} \times \frac{Poids(k, Can_1) + Poids(i, Can_2)}{2}$$

Le couple ayant le *coupleScore* le plus élevé est sélectionné comme sortie de la fonction de résolution de conflit.

Boucle de conflit. Au cours de la phase 1, une boucle de conflit peut se produire, c'est-à-dire que le candidat renvoyé par la fonction de résolution de conflit est lui-même mis en relation avec un autre itemset. Dans ce cas, nous continuons à faire appel à cette fonction jusqu'à ce que ses sorties ne soient pas déjà mises en relation. Si finalement on ne trouve pas de candidat pour un des itemsets en conflit, nous associons le candidat initial $Seq_2(j)$ avec l'itemsets en conflit le plus similaire et l'autre itemset reste sans correspondance.

Sortie de la phase 1. À la fin de cette phase, nous avons obtenu un ensemble de relations entre les itemsets des deux séquences. Ces liens sont conservés dans une liste nommée *mapOrder*. Nous mettons à la $i^{ème}$ place de cette liste le *timeStamp* de l'itemset de la séquence 2 mis en correspondance avec le $i^{ème}$ itemset de la séquence 1. Nous obtenons :

$$mapOrder = \{t_1, t_2, \dots, t_i, \dots, t_n\}$$

t_i est le *timeStamp* de l'itemset de la Seq_2 associé au $i^{ème}$ itemset de la Seq_1 .

Nous obtenons également le **score de mapping** c.-à-d. la moyenne de tous les poids des liens. Ce score prend en compte les mappings croisés. En effet, pour comparer $\langle(a)(b)\rangle$ avec $\langle(b)(a)\rangle$ et $\langle(a)(b)\rangle$ avec $\langle(b)(d)\rangle$ si l'on utilisait seulement les mappings respectant l'ordre (i.e. $(b) \rightarrow (b)$) nous traiterions $\langle(a)(b)\rangle$ avec $\langle(b)(a)\rangle$ de la même manière que $\langle(a)(b)\rangle$ avec $\langle(b)(d)\rangle$. Or, en considérant l'ensemble des poids, nous prenons en compte le poids de $(a) \rightarrow (a)$ (égal à 1) dans le premier cas et les itemsets sans mapping $(a), (d)$ dans le deuxième cas. Nous considérons donc ici l'influence des liens ne respectant pas l'ordre sur le degré de similarité.

Phase 2 – Calcul du score de l'ordre. En entrée de cette phase, nous avons les deux séquences et également la liste *mapOrder* sortie de la phase 1.

L'objectif de ce score est :

1. de trouver les liens respectant l'ordre des itemsets dans les deux séquences ;
2. de prendre en compte les positions des itemsets correspondants dans les deux séquences.

Pour cela, nous calculons deux scores : (1) *totalOrder* et (2) *positionOrder*.

totalOrder mesure le pourcentage de liens établis dans la phase 1 respectant l'ordre des itemsets (*l'exclusion des mappings croisés*). Dans la liste *mapOrder*, tant que les *timeStamps* augmentent, les liens correspondant respectent l'ordre. Nous cherchons donc toutes les sous-séquences croissantes et optimales de *mapOrder* pour trouver toutes les séries de liens qui sont conformes à l'ordre.

$$totalOrder = \frac{nbOrderedItemSets}{aveNbItemSets}$$

nbOrderedItemSets = le nombre d'itemsets dans la sous-séquence ordonnée

aveNbItemSets = la moyenne du nombre d'itemsets dans les deux séquences

Deuxièmement, nous mesurons la largeur entre les itemsets mis en relation en fonction de leur position avec le score *positionOrder*. Par exemple, sur la figure 3-(a), les itemsets sont mis en relation d'une manière plus proche que les item-

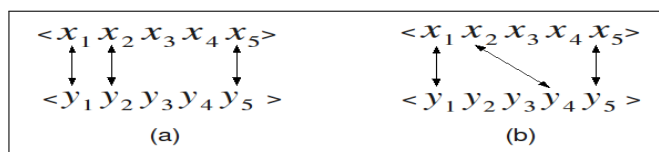


FIG. 3 – Distance entre mapping en fonction des positions des itemsets.

sets de la figure 3-(b). Nous utilisons de nouveau le *timeStamp* des itemsets pour obtenir cette information. *positionOrder* indique si la distance entre deux liens successifs dans la séquence 1 est égale à la distance entre ces mêmes liens dans la séquence 2. Sur la figure 3-(a), les liens sont à égale distance dans les deux séquences alors qu'ils ne le sont pas sur la figure 3-(b). *positionOrder* se calcule ainsi :

$$positionOrder = \sum_{i=1}^{|sub|} \left(\frac{|\sub(i) - \sub(i-1)| - (mapOrder^{-1}(\sub(i)) - mapOrder^{-1}(\sub(i-1)))|}{aveNbItemSets} \right)$$

sub(i) = la valeur de *i*^{ème} position dans la sous-séquence

mapOrder⁻¹(*x*) = la position de "*x*" dans *mapOrder*

Sortie de la phase 2 : Finalement, pour chaque sous-séquence croissante de *mapOrder*, nous multiplions *totalOrder* et *positionOrder* et nous gardons le score le plus élevé comme *score d'ordre* :

$$orderScore = \max\{totalOrder(sub) \times (1 - positionOrder(sub))\}$$

sub ∈ {sous_seqs croissantes et maximales du mapOrder}

Phase 3 – Calcul du degré de similarité. Nous calculons le degré de similarité *S²MP* par une agrégation entre le score d'ordre et le score de mapping avec une moyenne pondérée par des coefficients. Cela nous permet de moduler les scores selon les contextes d'application et la définition de la similitude.

$$S^2MP = \frac{(orderScore \times Co_1) + (mappingScore \times Co_2)}{Co_1 + Co_2}$$

Illustration. Soient les deux motifs $M_1 = \{(bc)(df)(e)\}$ et $M_2 = \{(abc)(mn)(de)(egh)(fg)\}$ que nous avons utilisés pour montrer les inconvénients de *LCS*.

Phase 1 : Calcul du score de mapping. Pour chaque itemset de $M_1(i)$, on cherche l'itemset le plus ressemblant dans $M_2(j)$ en calculant les poids.

$$\begin{aligned} - Poids(M_1(1), M_2(1)) &\implies Poids((bc), (abc)) = \frac{2}{3+2} = 0.8 \\ - Poids(M_1(1), M_2(2)) &\implies Poids((bc), (mn)) = \frac{0}{2+2} = 0 \\ - Poids(M_1(1), M_2(3)) &\implies Poids((bc), ((de))) = \frac{0}{2+2} = 0 \\ - Poids(M_1(1), M_2(4)) &\implies Poids((bc), ((egh))) = \frac{0}{2+3} = 0 \\ - Poids(M_1(1), M_2(5)) &\implies Poids((bc), ((fg))) = \frac{0}{2+2} = 0 \end{aligned}$$

On choisit le lien ayant le poids le plus élevé : $(bc) \rightarrow (abc)$. Nous effectuons les mêmes calculs pour les autres itemsets :

$Poids((df), (abc)) = \frac{0}{\frac{3+2}{2}} = 0$	$Poids((e), (abc)) = \frac{0}{\frac{3+2}{2}} = 0$
$Poids((df), (mn)) = \frac{0}{\frac{2+2}{2}} = 0$	$Poids((e), (mn)) = \frac{0}{\frac{2+2}{2}} = 0$
$Poids((df), ((de))) = \frac{1}{\frac{2+2}{2}} = 0.5$	$Poids((e), ((de))) = \frac{1}{\frac{1+2}{2}} = 0.6$
$Poids((df), ((egh))) = \frac{0}{\frac{2+3}{2}} = 0$	$Poids((e), ((egh))) = \frac{1}{\frac{1+3}{2}} = 0.5$
$Poids((df), ((fg))) = \frac{1}{\frac{2+2}{2}} = 0.5$	$Poids((e), ((fg))) = \frac{1}{\frac{2+2}{2}} = 0$

Dans le cas du 2^{ème} itemset de M_1 , les $poids((df), (de))$ et $poids((df), (fg))$ sont égaux. On sélectionne alors l'itemset ayant le timeStamp le plus petit (i.e. (de)) pour l'associer à l'itemset (df) . On a donc : $(df) \rightarrow (de)$.

Pour le 3^{ème} itemset, l'itemset sélectionné est (de) . Or, cet itemset a déjà été associé à l'itemset (df) de M_1 . Par conséquent, nous utilisons la fonction de résolution de conflit. Nous cherchons de nouveaux candidats dans M_2 pour les itemsets en conflit (df) et (e) avant et après l'itemset candidat actuel (de) . Nous obtenons les candidats suivants :

pour l'itemset (df) : $nextMaxBefore_{(df)} = \emptyset$ $nextMaxAfter_{(df)} = (fg)$	pour l'itemset (e) : $nextMaxBefore_{(e)} = \emptyset$ $nextMaxAfter_{(e)} = (egh)$
Les couples de mappings possibles : $\langle ((df), (de)), ((e), (egh)) \rangle$ $\langle ((e), (de)), ((df), (fg)) \rangle$.	

En utilisant les poids et en considérant les liens ne respectant pas l'ordre, nous obtenons :

$coupleScore(((df), (de)), ((e), (egh))) = \frac{0.5+0.5}{2} = 0.5$ $coupleScore(((e), (de)), ((df), (fg))) = \frac{1}{2} \times \frac{0.6+0.5}{2} = 0.27$

Nous choisissons le couple ayant le $coupleScore$ le plus élevé : $\langle ((df), (de)), ((e), (egh)) \rangle$. $(df) \rightarrow (de)$ et $(e) \rightarrow (egh)$.

Les mises en relation finales sont :

$\langle M_1(1) = (abc), M_2(1) = (ab) \rangle$ $\langle M_1(2) = (df), M_2(3) = (a) \rangle$ $\langle M_1(3) = (e), M_2(4) = (ca) \rangle$

Nous créons la liste $mapOrder$. $mapOrder(1) = 1$ car le timeStamp de l'itemset $M_2(1)$ mis en relation avec l'itemset $M_1(1)$ de la 1^{ère} séquence est égal à 1.

Nous obtenons : $mapOrder = \{1, 3, 4\}$

<p>Enfinement, le score de mapping est la moyenne des poids :</p> $mappingScore = \frac{poids((bc),(abc))+poids((df),(de))+poids((e),(egh))}{3} = 0.6$
--

Phase 2 – Calcul du score de l'ordre. Nous cherchons toutes les sous-séquences croissantes et optimales de la séquence $mapOrder$:

<p>La seule sous-séquence croissante maximale trouvée : $\{1, 3, 4\}$</p> <p>D'après les formules $totalOrder$ et $positionOrder$, le score de l'ordre se calcule ainsi :</p> $totalOrder((1,3,4)) = \frac{3}{(3+5)/2} = 0.75$ $positionOrder((1,3,4)) = \frac{ (3-1)-(2-1) }{(3+5)/2} + \frac{ (4-3)-(3-2) }{(3+5)/2} = 0.25$ $orderScore = 0.75 \times (1 - 0.25) = 0.56$
--

Phase 3 – Calcul de la similarité. Avec une multiplication entre le score d'ordre et le score de mapping, nous obtenons le degré de similarité des deux motifs séquentiels :

$S^2MP(M_1, M_2) = 0.56 \times 0.6 = 0.33$
--

4 Expérimentations

Une mesure de similarité doit saisir la ressemblance des objets comparés. Une telle mesure est généralement utilisée au sein d'un autre algorithme comme un clustering. Elle doit donc se calculer efficacement et passer à l'échelle. Nous expérimentons S^2MP avec deux objectifs principaux : (1) démontrer la qualité sémantique du degré de similarité obtenue par S^2MP , (2) mesurer l'efficacité de l'algorithme de S^2MP au niveau du temps d'exécution et de la taille mémoire utilisée.

Qualité sémantique de S^2MP . Nous testons S^2MP pour évaluer sa qualité sémantique et comparons les résultats obtenus par S^2MP et d'Edit distance.

Nous avons adapté l'algorithme de clustering k-means aux motifs séquentiels et aux deux mesures S^2MP et Edit distance. Nous avons appliqué ces deux versions de l'algorithme sur le même jeu de données constitué de 100 motifs séquentiels de tailles différentes. Nous utilisons Edit distance avec la différence ensembliste comme le coût de substitution (*i.e. la version utilisée dans ApproxMAP*). Les clusterings ont été réalisés jusqu'à ce que les clusters n'évoluent plus. Pour des applications comme le clustering où l'on a besoin d'une mesure symétrique, nous prenons la moyenne de S^2MP effectuée dans les deux directions pour rendre la mesure symétrique.

Pour comparer des clusters obtenus avec les deux mesures, nous ne pouvons pas utiliser les techniques de comparaison des clusterings (*mesures inter et intra-clusters*) effectuées à partir de la mesure de similarité (*ce que nous voulons évaluer*). L'algorithme de clustering est constant et ce sont les mesures qui diffèrent. La meilleure solution donc est de mesurer la pureté des clusters obtenus avec les deux mesures ou de comparer les résultats de deux clusterings avec un clustering de référence. Nous créons donc manuellement 10 groupes de motifs séquentiels similaires comme références à partir du jeu de données utilisé pour les clusterings. Parmi ces 10 groupes de référence, 4 groupes contiennent des motifs très différents des motifs séquentiels des autres groupes et 6 groupes contiennent des motifs qui ressemblent aux motifs d'un autre groupe. Cela nous permet d'évaluer la précision de chaque mesure lorsqu'il s'agit de distinguer les clusters avec une distance inter-cluster petite.

Nous calculons la précision et le rappel des clusters de chaque clustering par rapport aux références. Nous mesurons aussi l'entropie des clusters de chaque clustering qui permet de déterminer les groupes les plus homogènes (donc « meilleurs »). L'entropie moyenne de chaque clustering se calcule en prenant la moyenne des entropies de ses clusters.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
S^2MP	0.98	0	0.99	0.86	0.95	0	0.97	0.95	0.65	0
Edit distance	0.97	0	0.99	1.20	0.89	0	0.98	0.98	0.70	0.99

TAB. 1 – Entropie des clusters obtenus avec S^2MP et Edit distance.

Ces expérimentations montrent que les clusters obtenus avec S^2MP sont plus homogènes que ceux obtenus avec Edit distance. Le tableau 1 montre l'entropie moyenne obtenue avec chaque mesure. Plus l'entropie est petite plus l'ensemble est homogène. L'entropie moyenne de clustering avec S^2MP est 0.63 et en utilisant Edit distance est égale à 0.77.

Le tableau 2 montre la précision et le rappel des clusters obtenus avec chaque mesure.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Précision S^2MP	0,57	1	0,53	0,71	0,65	1	0,5	0,6	0,68	1
Rappel S^2MP	0,4	1	0,7	1	1	0,5	0,5	0,4	1	0,4
Précision Edit	0,6	1	0,53	0,58	0,68	1	0,4	0,62	0,83	0,57
Rappel Edit	0,3	1	0,6	1	0,9	0,8	0,2	0,5	1	0,4

TAB. 2 – Précision et rappel des clusters obtenus avec S^2MP et Edit distance.

Nous expérimentons également S^2MP et Edit distance sur leur capacité à identifier des motifs séquentiels similaires dans des contextes différents. Nous considérons le contexte des données bio-informatiques et les caractéristiques particulières des motifs séquentiels extraits à partir de puces ADN. Dans ce domaine, d'après les experts, les contenus des itemsets (*i.e. items*) sont plus importants que l'ordre des itemsets lorsqu'il s'agit de comparer deux motifs. Par exemple, les deux motifs $M_1 = \{(a)(b)(c)\}$ et $M_2 = \{(b)(a)(c)\}$ sont plus ou moins similaires car les contenus des itemsets se ressemblent même si l'ordre n'est pas respecté.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
S^2MP	0,99	0,52	0,99	0,99	0,99	0	0	0,99	0	0,98
Edit distance	1,2	1,2	1,3	1,3	1,3	1,2	1,4	0,95	0,4	1,7

TAB. 3 – Entropie des clusters obtenus avec S^2MP et Edit distance quand le contenu des itemsets est plus important que leur ordre – (e.g. motifs extraits issus de puce ADN).

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Précision S^2MP	0,55	0,71	0,54	0,54	0,55	1	1	0,53	1	0,57
Rappel S^2MP	0,5	1	0,6	0,6	0,5	0,6	1	0,7	1	0,4
Précision Edit	0,52	0,61	0,46	0,60	0,50	0,5	0,5	0,37	0,9	0,16
Rappel Edit	0,9	0,6	0,6	0,3	0,5	0,6	0,7	0,3	0,9	0,2

TAB. 4 – Précision et rappel des clusters obtenus avec S^2MP et Edit distance quand le contenu des itemsets importe plus que leur ordre.

Pour comparer S^2MP et Edit distance dans telle situation, nous créons manuellement 10 groupes de 10 motifs similaires au niveau du contenu des itemsets mais ayant des ordres différents. Ces groupes ont une distance inter-groupe petite. Nous effectuons un clustering sur ce dernier jeu de données avec S^2MP en choisissant le deux comme coefficient du score de mapping et un comme coefficient du score d'ordre (i.e. nous configurons S^2MP pour que les contenus des itemsets soient considérés comme plus importants que l'ordre des itemsets). L'idée n'est pas de trouver des poids idéaux qui peuvent être déterminés par les expérimentations.

Ensuite, nous réalisons un clustering sur ce jeu de données avec Edit distance. Les résultats montrent que S^2MP permet de mieux identifier les motifs similaires dans un contexte d'application particulier tel que les motifs séquentiels issus de données transcriptomiques. Cela prouve que S^2MP est bien paramétrable est adaptable à différents contextes. Les résultats obtenus avec Edit distance dans ce contexte, ne sont pas satisfaisant. Le tableau 3 souligne la pertinence de S^2MP dans ce contexte en fonction d'entropie des clusters obtenus. Sur ce jeu de données, l'entropie moyenne de clustering en utilisant S^2MP est 0.64 et en utilisant Edit distance est 1.19. Le tableau 4 montre la précision et le rappel des clusters obtenus avec S^2MP et Edit distance sur ce jeu de données.

Efficacité de S^2MP .

Nous montrons ici que notre mesure de similarité est très efficace au niveau du temps d'exécution et de la taille de mémoire utilisée. Nous expérimentons l'efficacité de S^2MP selon trois paramètres : (1) le nombre d'items dans les séquences, (2) le nombre itemsets et (3) le nombre de séquences dans la base.

Nous créons une matrice de similarité Mat_{sim} de dimension $(n \times n)$ où n représente le nombre de motifs séquentiels dans la base. S^2MP n'étant pas symétrique, nous calculons la totalité de $Mat_{sim}(n, n)$. Le temps nécessaire pour remplir $Mat_{sim}(n, n)$ est le temps nécessaire aux $n \times n$ comparaisons de similarité. Les expérimentations sont effectuées sur une machine ayant un CPU Intel 2GHz avec 2Go de mémoire vive. La mesure de similarité est développée en Java 5. Nous avons utilisé ici deux types de jeu de données : (1) des motifs séquentiels fréquents (2) des séquences de données. Les motifs séquentiels fréquents sont extraits à partir de données synthétiques générées par le générateur de données IBM quest⁵. Pour vérifier l'influence des conflits sur le temps d'exécution, nous avons fait un test avec des séquences de données car elles permettent d'obtenir de nombreux conflits lors de la mise en relation des itemsets.

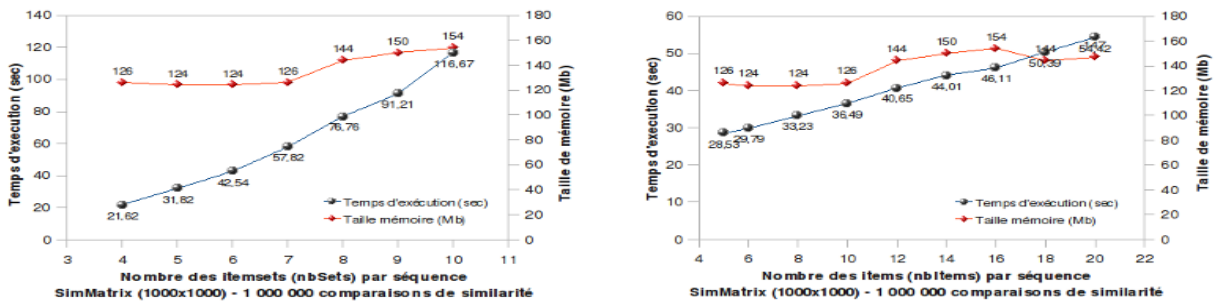


FIG. 4 – Temps de calcul et taille de mémoire en fonction du nombre d'itemsets (gauche) et d'items (droit).

⁵www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html

Sur les tests en fonction du nombre d'itemsets et d'items, les jeux de données sont constitués de 1 000 motifs séquentiels (*c.-à-d. nous réalisons 1,000,000 de comparaisons de similarité*).

Résultats d'expérimentation sur l'efficacité de S²MP. La figure 4 représente l'évolution du temps de calcul des 1,000,000 comparaisons et la taille de mémoire utilisée par rapport au nombre d'itemsets (gauche) et d'items (droit) par séquence. Selon les courbes, la taille de mémoire utilisée ne change pas considérablement. Le temps de calcul de $Mat_{sim}(1^M, 1^M)$ quand il y a 10 itemsets par séquence est satisfaisant (116 secs). Notre expérimentation montre que le nombre d'items par séquence n'influence pas le temps de comparaison de la similarité autant que le nombre d'itemsets par séquence. Sur la figure 5, nous montrons le temps de calcul de $Mat_{sim}(n, n)$ (figure de gauche) et la taille mémoire

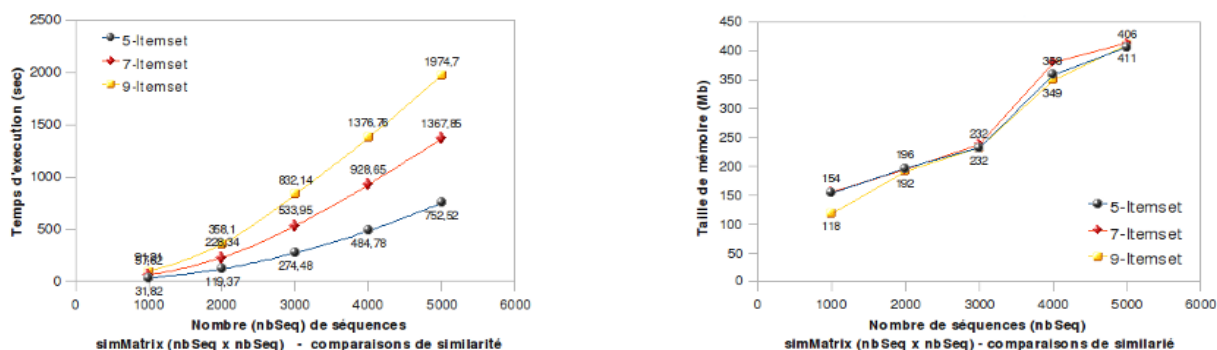


FIG. 5 – Temps de calcul de $Mat_{sim}(n, n)$ (gauche) et taille de mémoire utilisée (droit) en fonction du nombre de séquences.

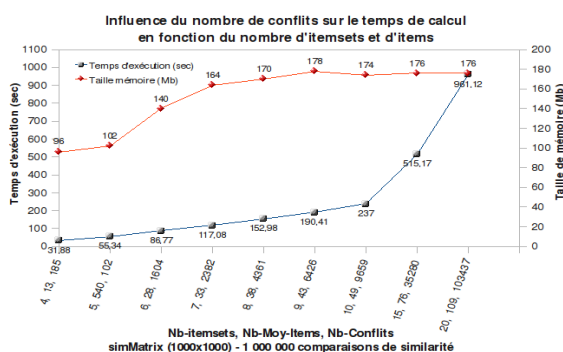


FIG. 6 – Influence des conflits sur le temps de calcul de la mesure de similarité

utilisée (figure de droite) quand le nombre de séquences augmente. Dans chaque cas, il y a $n \times n$ comparaisons de similarité où n est le nombre de séquences dans le jeu de données. Nous réalisons ce test sur des séquences de 5, 7 et 9 itemsets. Dans le cas, où il y a 5 000 séquences (*c.-à-d. 25,000,000 de comparaisons de similarité*) et que chaque séquence contient 9 itemsets, le temps d'exécution est seulement de 1974 secs.

Sur la figure 6, nous montrons le résultat de l'expérimentation sur les séquences de données. Pour chaque cas, nous avons noté le nombre de conflits résolus lors du calcul de $Mat_{sim}(1^M, 1^M)$. L'axe X représente les différents jeux de données. Pour chacun, le nombre d'itemsets et le nombre moyen d'items par séquences sont notés. Il existe 1 000 séquences dans chaque jeu de données (1^M comparaisons de similarité). Les courbes représentent le temps de remplissage de $Mat_{sim}(1^M, 1^M)$ pour chaque cas et la taille de mémoire utilisé. Dans le cas où il y a 20 itemsets et en moyenne 109 items par séquence, 103 437 conflits sont résolus et le temps d'exécution de 1^M calcul de similarité est égal à 961 secs.

5 Conclusion

Dans cet article, nous avons défini une mesure de similarité (S²MP) adaptée aux motifs séquentiels prenant en compte toutes leurs caractéristiques et notamment leur sémantique. Le degré de similarité est composé de deux scores. Ces scores mesurent la similarité des motifs séquentiels à la fois au niveau de l'ordre des itemsets et de leurs positions dans les

séquences (*score d'ordre*) mais aussi au niveau des items contenus dans les itemsets correspondant (*score de mapping*). La combinaison de deux scores indépendants permet d'avoir une mesure modulaire. Elle est donc adaptable selon le contexte et le sens des itemsets dans le domaine en modifiant le score concerné. S²MP surmonte les inconvénients des mesures classiques comme *LCS* et *Edit distance* pour les motifs séquentiels. Les expérimentations montrent que S²MP se calcule très rapidement même lorsque le nombre de séquences ayant plusieurs itemsets est élevé. Plusieurs domaines et méthodes comme le clustering de motifs séquentiels, la détection d'outliers, l'extraction de motifs séquentiels sous contrainte de similarité, la compression des motifs séquentiels, la visualisation des motifs similaires, etc. sont envisageables comme applications de S²MP.

Références

- Agrawal, R., C. Faloutsos, et A. N. Swami (1993). Efficient similarity search in sequence databases. In D. Lomet (Ed.), *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, Chicago, Illinois, pp. 69–84. Springer Verlag.
- Agrawal, R. et R. Srikant (1995). Mining sequential patterns. In P. S. Yu et A. S. P. Chen (Eds.), *Eleventh International Conference on Data Engineering*, Taipei, Taiwan, pp. 3–14. IEEE Computer Society Press.
- Bozkaya, T., N. Yazdani, et Z. M. Ozsoyoglu (1997). Matching and indexing sequences of different lengths. In *CIKM*, pp. 128–135.
- Capelle, M., C. Masson, et J.-F. Boulicaut (2002). Mining frequent sequential patterns under a similarity constraint. In *IDEAL*, pp. 1–6.
- Garofalakis, M. N., R. Rastogi, et K. Shim (1999). SPIRIT : Sequential pattern mining with regular expression constraints. In *The VLDB Journal*, pp. 223–234.
- Guralnik, V. et G. Karypis (2001). A scalable algorithm for clustering sequential data. In *ICDM '01 : Proceedings of the 2001 IEEE International Conference on Data Mining*, Washington, DC, USA, pp. 179–186. IEEE Computer Society.
- Hartigans, J. (1975). *Clustering Algorithms*. John Wiley and Sons, Inc.
- Jianhua Zhu, Z. W. (2005). Fast : A novel protein structure alignment algorithm. Volume 58, Bioinformatics Program, Boston University, Boston, Massachusetts ; Biomedical Engineering Department, Boston University, Boston, Massachusetts.
- Kum, H.-C. (2004). *Approximate Mining of Consensus Sequential Patterns*. Ph. D. thesis, University of North Carolina.
- Kum, H.-C., J. Pei, W. Wang, et D. Duncan (2003). Approxmap : Approximate mining of consensus sequential patterns. In *SDM*.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8.
- Mannila, H. et P. Ronkainen (1997). Similarity of event sequences. In *TIME '97 : Proceedings of the 4th International Workshop on Temporal Representation and Reasoning (TIME '97)*, Washington, DC, USA, pp. 136. IEEE Computer Society.
- Moen, P. (2000). *Attributes, Event Sequence, and Event Type Similarity Notions for Data Mining*. Ph. D. thesis, University of Helsinki, Finland.
- Morzy, T., M. Wojciechowski, et M. Zakrzewicz (1999). Pattern-oriented hierarchical clustering. In *Advances in Databases and Information Systems*, pp. 179–190.
- Pei, J., J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, et M. chun Hsu (2001). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. pp. 215–224.
- Plantevit, M., S. Goutier, F. Guisnel, A. Laurent, et M. Teisseire (2007). Mining unexpected multidimensional rules. In *DOLAP '07 : Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pp. 89–96.
- Sequeira, K. et M. J. Zaki (2002). Admit : anomaly-based data mining for intrusions. In *KDD*, pp. 386–395.
- Tversky, A. (1977). *Psychological Review* (84), 327–352.

Summary

In the field of knowledge extraction, comparing the similarity of objects is an essential task, for example to identify regularities or to build homogeneous clusters of objects. In the case of sequential data seen in various fields of application (e.g. series of customers purchases, Internet navigation) this problem (i.e. comparing the similarity of sequences) is very important. There are already some similarity measures as Edit distance and LCS suited to simple sequences, but these measures are not relevant in the case of complex sequences composed of sets of items, as is the case of sequential patterns. In this paper we propose a new similarity measure taking the characteristics of sequential patterns into account. S²MP is an adjustable measure depending on the importance given to each characteristic of sequential patterns according to context, which is not the case of existing measures. Our measure has been validated by experiments on different data sets.