Découverte complète et interactive de motifs temporels avec contraintes numériques à partir de séquences d'événements.

Damien Cram*, Béatrice Fuchs*, Yannick Prié*, Alain Mille*

* Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
damien.cram,beatrice.fuchs,yannick.prie,alain.mille@liris.cnrs.fr
http://liris.cnrs.fr

Résumé. Nous proposons une méthode complète d'extraction de motifs temporels fréquents avec contraintes numériques à partir d'une séquence d'événements. Ces motifs, appelés *chroniques*, sont très expressifs par rapport aux motifs séquentiels plus «classiques» tels que les épisodes parallèles ou en série, ce qui implique une grande complexité dans la résolution complète. Un algorithme de résolution permettant l'introduction d'heuristiques et de contraintes utilisateur est proposé. Il est ensuite discuté comment mettre à profit les interactions avec l'utilisateur pour compenser sa grande complexité et le rendre praticable.

1 Introduction : contexte et motivations

[1]De nombreux domaines génèrent des données à caractère temporel et nécessitent des méthodes de fouilles adaptées à cet aspect. C'est notamment le cas du domaine de l'analyse de l'activité, particulièrement lorsque l'activité a été capturée et modélisée en une trace d'interactions racontant l'histoire des actions du sujet (Cram et al., 2008). C'est dans ce cadre que Cram et al. (2008) ont besoin d'une méthode permettant l'abstraction des connaissances sur l'usage d'un système par son utilisateur. Pour cela, il est proposé de trouver dans la trace des motifs temporels pertinents, pouvant être considérés comme la description abstraite d'une situation.

Dans cet article nous proposons une formalisation de ce problème et nous en proposons une solution. La trace d'interactions est considérée comme une séquence d'événements où chaque événement représente une action du sujet. Les méthodes *Winepi* et *Minepi* (Mannila et al., 1997) extraient dans une séquence les épisodes fréquents, en série ou en parallèle. De nombreuses améliorations ont été ajoutées à cette méthode, mais rares sont celles qui ont porté sur la découverte d'épisodes «hybrides», c'est-à-dire imposant un ordre partiel sur les occurrences d'événements. Rares sont également les méthodes qui proposent de quantifier les écarts entre deux événements d'un même épisode.

Ces deux limitations sont très préjudiciables à la bonne représentation de certains motifs de comportement dans l'activité. Par exemple, dans l'activité d'achat en ligne, le motif «renseigner son adresse» serait constitué des événements «cliquer sur le lien `Mon_adresse`», *puis* « remplir le champ `Numéro`» et « remplir le champ `Nom_de_rue`» dans n'importe quel ordre, *puis* de « cliquer sur `Valider`». Sans un formalisme hybride, il serait impossible de représenter le parallélisme des deux événements «remplir» et leur sérialité avec les deux autres. D'autre part, la quantification des écarts entre événements dans le motif permet de savoir si les actions ont été réalisées rapidement ou non, puis par comparaison entre utilisateurs de savoir si le sujet est débutant ou expérimenté, et ainsi d'adapter l'assistance.

Le formalisme de *chronique* (Dousson et al., 1993) permet de représenter ces deux aspects. Une chronique est un épisode permettant de spécifier des écarts minimaux et maximaux entre chaque événement de l'épisode. La question de la découverte de chroniques dans une séquence d'événements a été traitée par Duong (2001), mais la méthode proposée n'est pas complète, car une seule contrainte temporelle n'est extractible pour un couple d'événements donné (cf. section 2). Cette limitation est très restrictive dans la mesure où la grande majorité des chroniques ne seront pas candidates dans le processus de découverte. De plus, dans le cadre de la recherche de motifs dans l'activité, cette limitation nous semble arbitraire.

Afin de ne pas restreindre ainsi *a priori* la découverte de chroniques intéressantes à un sous-ensemble de l'ensemble des chroniques candidates, nous proposons une méthode complète de découverte de chroniques. Évidemment la contrainte de la complétude dans un processus de découverte de motifs très expressifs comme les chroniques fait exploser la complexité, comme nous le verrons par la suite. Mais nous pensons qu'il existe des stratégies pour rendre une telle méthode praticable.

[1]Cette recherche est financée par l'ANR dans le cadre du projet PROCOGEC (`www.procogec.com`).

En effet, la découverte de connaissances est un processus interactif et itératif. Une fois qu'un algorithme de fouille de données est exécuté, ses résultats sont analysés par l'utilisateur qui, selon sa satisfaction, ajustera plus ou moins les paramètres d'entrée de l'algorithme dans l'espoir d'obtenir de meilleurs résultats. Il existe tout de même un certain nombre de travaux qui tentent de bénéficier des interactions avec l'utilisateur pour améliorer le processus de découverte, notamment en ce qui concerne les séquences d'événements. Parthasarathy et al. (1999) et Lin et Lee (2004) proposent de stocker et d'organiser en mémoire, appelée parfois base de connaissances, tout ou partie des résultats des exécutions précédentes de telle sorte que lorsqu'une nouvelle requête est soumise, il soit possible de calculer les nouveaux résultats à partir des informations déjà présentes en mémoire sans avoir à parcourir à nouveau l'ensemble des données. Ces méthodes sont qualifiées d'«interactives», mais l'unique but des interactions est l'optimisation du temps d'exécution du processus au fur et à mesure que les requêtes s'accumulent.

D'autres travaux plus récents visent à tirer profit des interactions avec l'utilisateur pour bâtir et maintenir une base de connaissances que l'utilisateur a sur son domaine. Les algorithmes utilisent ensuite ces connaissances pour mesurer l'intérêt des différents motifs du point de vue de l'utilisateur. Par exemple, Fauré (2007) a mis en place un réseau bayésien pour modéliser les dépendances connues de l'utilisateur sur son domaine. Ce réseau bayésien est mis à jour par les informations annotées par l'utilisateur sur les règles extraites par l'algorithme.

L'interactivité dans le processus de découverte permet notamment l'accélération de l'exécution et l'augmentation de la pertinence des motifs extraits. L'activité fondamentale en fouille de données consiste à élaborer des algorithmes permettant d'explorer le plus grand volume de données possible en un temps le plus acceptable possible. Mais grâce à ces deux aspects de l'interactivité, il est possible de s'attaquer à des problèmes de découverte de plus grande complexité. C'est ce que nous faisons ici. L'objectif principal de cet article n'est pas d'étudier cet aspect interactif, mais de proposer une méthode de fouille répondant au problème posé par notre domaine d'application qui s'avère être très coûteux en calcul. La méthode proposée a été pensée pour offrir des mécanismes d'optimisation par les interactions, dans des développements futurs.

La section 2 pose le problème de la découverte complète de chroniques et une méthode de résolution est proposée en section 3. Les optimisations interactives possibles sont discutées en section 4. La section 5 conclut et dresse les perspectives de ce travail.

2 Formulation du problème

Une *séquence d'événements* est un ensemble noté $\mathcal{S} = \langle (e_1, t_1) \dots (e_l, t_l) \rangle$, où chaque (e_i, t_i) est un *événement* composé d'un *type d'événement* $e_i \in \mathbb{E}$ et d'un entier t_i appelé sa *date*. L'ensemble $\mathbb{E}$ est supposé totalement ordonné et fini. Les motifs temporels recherchés sont les *chroniques*. Une *contrainte temporelle* est un quadruplet (e_g, e_d, I^-, I^+), noté $e_g[I^-, I^+]e_d$, où $(e_g, e_d) \in \mathbb{E}^2$ et I^- et I^+ sont deux entiers vérifiant $I^- \leq I^+$. On dit que deux événements (e_1, t_1) et (e_2, t_2) satisfont la contrainte $e_g[I^-, I^+]e_d$ si et seulement si $e_1 = e_g$ et $e_2 = e_d$ et $t_2 - t_1 \in [I^-, I^+]$, ou si $e_1 = e_d$ et $e_2 = e_g$ et $t_1 - t_2 \in [I^-, I^+]$. Une *chronique* est un couple $(\mathcal{E}, \mathcal{T})$, où :

- $\mathcal{E} = \varepsilon_1 \dots \varepsilon_n$ avec $\forall i, \varepsilon_i \in \mathbb{E}$ et $\forall i < j, \varepsilon_i \leq_{\mathbb{E}} \varepsilon_j$; ($\mathcal{E}$ est la partie *épisode* de la chronique ; n est la *taille* de la chronique)
- $\mathcal{T} = \{\tau_{ij}\}_{1 \leq i < j \leq |\mathcal{E}|}$ est un ensemble de contraintes temporelles sur $\mathcal{E}$ tel que $\forall i < j, \tau_{ij} = \varepsilon_i[\tau_{ij}^-, \tau_{ij}^+]\varepsilon_j$. ($\mathcal{T}$ est la partie *contraintes* de la chronique)

Enfin, une *occurrence* d'une chronique $\mathcal{C} = (\mathcal{E}, \mathcal{T})$ est une liste d'événements de $\mathcal{S}$, notée $\langle (\varepsilon_1, t_1) \dots (\varepsilon_n, t_n) \rangle$ et vérifiant $\forall i < j, t_j - t_i \in [\tau_{ij}^-, \tau_{ij}^+]$. On notera $\mathcal{O}(\mathcal{C}, \mathcal{S})$ l'ensemble des occurrences de $\mathcal{C}$ dans $\mathcal{S}$.

Par exemple, avec $\mathbb{E} = A, B, C$ et $\mathcal{S} = \langle (A,1)(C,2)(B,4)(A,5)(C,5)(B,6) \rangle$, les occurrences de $\mathcal{C}_1 = (ABC, \{A[1,3]B, A[1,1]C, B[-2,-1]C\})$ sont $\langle (A,1)(C,2)(B,4) \rangle$ et $\langle (A,5)(C,5)(B,6) \rangle$ (cf. figure 1).

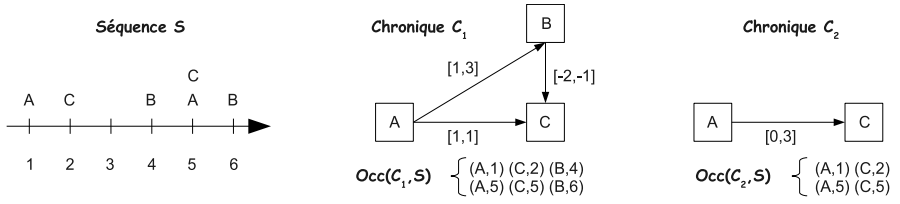

FIG. 1 – *Exemple de séquence d'événements, de chroniques et d'inclusion ($\mathcal{C}_1 \preceq \mathcal{C}_2$).*

Une chronique $\mathcal{C}$ est dite *incluse* dans une chronique $\mathcal{C}'$ ($\mathcal{C}'$ est *plus générale* que $\mathcal{C}$) si $\mathcal{E}'$ est un sous-épisode de $\mathcal{E}$, et si avec h la *fonction d'inclusion* telle que $\mathcal{E}' = \varepsilon'_1 \dots \varepsilon'_{n'} = \varepsilon_{h(1)} \dots \varepsilon_{h(n')}$ on a $\forall i < j, [\tau_{ij}^-, \tau_{ij}^+] \subseteq [\tau_{h(i)h(j)}^-, \tau_{h(i)h(j)}^+]$. Par exemple, sur la figure 1, $\mathcal{C}_1$ est incluse dans $\mathcal{C}_2$.

Étant donné un entier f_{seuil} appelé la *fréquence seuil*, on définit la *fréquence* d'une chronique $\mathcal{C}$ dans une séquence $\mathcal{S}$ comme le nombre de ses occurrences : $f(\mathcal{C}) = |\mathcal{O}(\mathcal{C}, \mathcal{S})|$. Duong (2001) propose de définir $\mathcal{O}(\mathcal{C}, \mathcal{S})$ comme l'ensemble des occurrences de $\mathcal{C}$ reconnues par l'algorithme *CRS* de Dousson et al. (1993). *CRS* ne reconnaît qu'un ensemble d'occurrences $\mathcal{O}(\mathcal{C}, \mathcal{S})_{CRS}$ tel qu'il n'existe pas deux occurrences partageant le même événement. C'est la définition que nous garderons pour la suite, car elle donne un nombre d'occurrences intuitif et présente des propriétés intéressantes pour la découverte, notamment la monotonie par rapport à $\preceq$ (Duong, 2001).

Étant donnés une séquence $\mathcal{S}$ et un minimum de fréquence f_{seuil}, il existe pour chaque épisode $\varepsilon_1\varepsilon_2$ plusieurs contraintes temporelles τ telles que la chronique $(\varepsilon_1\varepsilon_2, \{\tau\})$ soit fréquente (i.e. $f(\varepsilon_1\varepsilon_2, \{\tau\}) \geq f_{seuil}$). Pour trouver toutes ces contraintes τ, on utilise une méthode inspirée de celle proposée par Duong (2001). Reprenons l'exemple de $\varepsilon_1\varepsilon_2 = AB$ avec $f_{seuil} = 3$. On calcule d'abord l'ensemble des amplitudes des occurrences de AB. Cet ensemble est $\{3, 5, -1, 1\}$ et on le tri en $\{-1, 1, 3, 5\}$. À partir de cet ensemble, on bâtit tous les intervalles $[i, j]$ assurant que $f(A[i, j]B) \geq 3$. Les intervalles tels que $f(A[i, j]B) = 3$ sont $[-1, 3]$ et $[1, 5]$, et celui pour $f(A[i, j]B) = 4$ est $[-1, 5]$. On peut organiser ces trois intervalles en un graphe d'inclusions de contraintes pour AB. En appliquant cette méthode pour BC et AC, on obtient ainsi pour $\mathcal{S}$ et $f_{seuil} = 3$ une base de contraintes $\mathcal{D}_0$ (cf. figure 2).

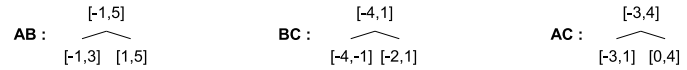

FIG. 2 – *Base de contrainte $\mathcal{D}_0$ obtenue à partir de $\mathcal{S}$ pour $f_{seuil} = 3$.*

On notera $\mathcal{D}^\top$ l'ensemble des chroniques de taille 2 qui sont construites à partir de la contrainte la plus générale de $\mathcal{D}$. Ici, $\mathcal{D}^\top = \{A[-1, 5]B, B[-4, 1]C, A[-3, 4]C\}$.

Le problème de la découverte complète de chroniques fréquentes ne peut se formuler qu'une fois que $\mathcal{D}$ a été ainsi construite : « trouver toutes les chroniques dont les contraintes temporelles sont dans $\mathcal{D}$ précédemment construite, telles que $f(\mathcal{D}) \geq f_{seuil}$ ».

La méthode non-complète de Duong (2001) propose pour chaque couple d'événements de ne garder dans la base de contraintes qu'une seule contrainte numérique et non pas l'ensemble des contraintes fréquentes. Ce choix est fait par l'intermédiaire d'un mécanisme de notation de chaque contrainte numérique. En conséquence, l'ensemble des chroniques découvertes par la suite étant constitué uniquement de chroniques construites à partir de cette base incomplète de contraintes n'est pas complet.

3 Résolution complète

Étant donnée une base de contraintes $\mathcal{D}$, une chronique $\mathcal{C}'$ est dite *directement incluse* dans une chronique $\mathcal{C}$ si et seulement si $\mathcal{C}$ et $\mathcal{C}'$ sont contruites à partir de $\mathcal{D}$, et $\mathcal{C} \prec \mathcal{C}'$ (inclusion stricte), et il n'existe pas de chroniques $\mathcal{C}''$ construite à partir de $\mathcal{D}$ telle que $\mathcal{C} \prec \mathcal{C}'' \prec \mathcal{C}'$. Ce concept d'*inclusion directe* permet d'introduire la notion de *successeur* d'une chronique, où $\mathcal{C}$ est dite *successeur* de $\mathcal{C}'$ si $\mathcal{C}$ est directement incluse dans $\mathcal{C}'$, relativement à $\mathcal{D}$. C'est cette notion de *successeur* qui est à la base de la résolution complète. En effet, à partir d'un petit ensemble de chroniques très générales, la résolution consiste à générer et tester pour chacune d'elle et en profondeur d'abord de successeurs en successeurs toutes les chroniques candidates, jusqu'à être sûr qu'il n'y ait plus de successeurs qui puissent être fréquents.

Il existe deux types d'opérateur permettant de générer un successeur d'une chronique $\mathcal{C}$. Le premier consiste à ajouter un type d'événement à $\mathcal{C}$, le deuxième consiste à sélectionner dans $\mathcal{C}$ une contrainte $\tau_{ij} = e_i[\tau_{ij}^-, \tau_{ij}^+]e_j$ et à la remplacer par $\sigma_{ij} = e_i[\sigma_{ij}^-, \sigma_{ij}^+]e_j$, où $\sigma_{ij} \in \mathcal{D}$ et σ_{ij} est directement plus stricte que τ_{ij} dans $\mathcal{D}$. Le nombre de successeurs d'une chronique relativement à une base de contraintes est donc fini puisqu'il y a $|\mathbb{E}|$ opérateurs du premier type et en notant $\mathcal{T}$ l'ensemble des contraintes de $\mathcal{C}$, $\sum_{\tau_{ij} \in \mathcal{T}} n_{\tau_{ij}}$ opérateurs du deuxième type, où $n_{\tau_{ij}}$ est le nombre de contraintes directement plus strictes que τ_{ij} dans $\mathcal{D}$.

Pour résoudre le problème de découverte complète, on applique l'algorithme suivant.

Entrées: $\mathcal{S}$; $\mathcal{D}$; f_{seuil}
Sorties: F
1: F$\leftarrow \emptyset$
2: DéjàTraitées$\leftarrow \emptyset$
3: O $\leftarrow \mathcal{D}^\top$
4: **répéter**
5: $\quad\mathcal{C} \leftarrow$ choisir(O)
6: $\quad$Ajouter $\mathcal{C}$ à DéjàTraitées
7: $\quad$**si** a_plus_gén(nonF,$\mathcal{C}$) **alors**
8: $\quad\quad$continuer à la ligne 4
9: $\quad$**si** $\neg$a_plus_stricte(F,$\mathcal{C}$) **alors**
10: $\quad\quad f(\mathcal{C}) \leftarrow$compter($\mathcal{C}, \mathcal{S}$)
11: $\quad\quad$**si** $f(\mathcal{C}) \geq f_{seuil}$ **alors**
12: $\quad\quad\quad$ajout_min($\mathcal{C}$,F)

13: **sinon**
14: ajout_max($\mathcal{C}$,nonF)
15: continuer à la ligne 4
16: **sinon**
17: $Succ(\mathcal{C}) \leftarrow$ gén_succ($\mathcal{C}$)
18: **pour** chaque $\mathcal{C}' \in Succ(\mathcal{C})$ **faire**
19: **si** est_acceptable($\mathcal{C}'$,DéjàTraitées) **alors**
20: O $\leftarrow$ O $\cup \{\mathcal{C}'\}$
21: **jusqu'à** O $\neq \emptyset$
22: **retourner** F

L'idée de l'algorithme est de prendre chaque chronique $\mathcal{C}$ construite à partir de $\mathcal{D}$, de calculer sa fréquence dans $\mathcal{S}$ (10) et si on a $f(\mathcal{C}) \geq f_{seuil}$ de traiter de même tous ses successeurs par la suite. Ce processus est effectué dans la boucle 4–21. L'ensemble O qui contient à tout moment l'ensemble des chroniques à traiter dans les itérations futures. L'ensemble F contient l'ensemble des chroniques fréquentes minimales, c'est-à-dire qu'il n'existe pas dans F $\mathcal{C}$ et $\mathcal{C}'$ telles que $\mathcal{C} \prec \mathcal{C}'$. À chaque itération on prend une nouvelle chronique $\mathcal{C}$ de O à traiter (6). Si $\mathcal{C}$ est fréquente, on génère alors ses successeurs et on les ajoute à l'ensemble O (17–20). Le processus se termine lorsqu'il n'y a plus de chroniques à traiter (21) et on retourne alors F (22). Il est initié avec toutes les contraintes les plus générales de $\mathcal{D}$ (3) et on retourne F (22).

Les autres lignes de l'algorithme correspondent à deux optimisations de ce processus. La première optimisation vise à ne compter la chronique que si on n'est pas déjà sûr qu'elle sera fréquente ou non-fréquente (tests des lignes 7 et 9). À cette fin, un ensemble nonF contenant toutes les chroniques maximales non-fréquentes est maintenu tout au long de la découverte. Si nonF contient une chronique plus générale que $\mathcal{C}$, alors cela signifie que sans avoir à compter $\mathcal{C}$, $\mathcal{C}$ n'est pas fréquente car f est monotone. La procédure a_plus_gén renverra alors vrai et la processus reprendra à la ligne 4. De même, si $\mathcal{C}$ est plus générale qu'une chronique déjà fréquente, alors ce n'est la peine ni de la compter, ni de traiter ses successeurs (cf. ligne 9). La deuxième consiste à vérifier que les successeurs sont acceptables (19). Une chronique est dite acceptable s'il n'existe pas une chronique préalablement traitée qui est non-fréquente et plus générale et si elle n'a elle-même pas été préalablement traitée :

Entrées: $\mathcal{C}$, DéjàTraitées
1: **si** [a_plus_gén(nonF,$\mathcal{C}$)] ou [contient(DéjàTraitées,$\mathcal{C}$)] **alors**
2: **retourner faux**
3: **retourner vrai**

C'est également cette méthode est_acceptable qui permet de pousser les contraintes utilisateurs monotones dans le processus de découverte de chroniques, par l'ajout de la condition $[\mathcal{C}\ ne\ satifait\ pas\ les\ contraintes\ monotones]$.

4 Analyse de l'agorithme et premiers résultats

4.1 Propriétés de l'algorithme

La complétude de l'algorithme proposé peut être démontrée par récurrence sur la *profondeur* de la chronique. La *profondeur* d'une chronique est le nombre d'opérateurs qu'il faut appliquer à une chronique de $\mathcal{D}^\top$ pour pouvoir l'obtenir. Par exemple dans $\mathcal{D}_0$ la chronique $\mathcal{C}_3 = (ABC, \{A[-1,3]B, A[-3,4]C, B[-2,1]C\})$ s'obtient à partir de $(AB, \{A[-1,5]B\}) \in \mathcal{D}_0^\top$ par application de trois opérateurs :

1. l'ajout de l'événement C à $(AB, \{A[-1,5]B\})$, qui devient $(ABC, \{A[-1,5]B, A[-3,4]C, B[-4,1]C\})$,
2. le renforcement de $A[-1,5]B$ en $A[-1,3]B$ par $\mathcal{D}_0$,
3. le renforcement de $B[-4,1]C$ en $B[-2,1]C$ par $\mathcal{D}_0$.

On a ainsi $profondeur(\mathcal{C}_3) = 3$. On peut ensuite démontrer pour tout $n > 2$ le prédicat suivant :

> $P(n)$: si $\mathcal{C}$ est une chronique fréquente de profondeur n construite à partir de $\mathcal{D}$, alors $\mathcal{C}$ sera ajoutée une et une seule fois à O.

$P(0)$ est évident puisque l'ensemble des chronique de profondeur 0 est $\mathcal{D}^\top$ et que cet ensemble est ajouté à O lors de l'initialisation de l'algorithme (cf. ligne 3). Si on suppose $P(n)$ vrai, alors $P(n+1)$ s'avère vrai par le raisonnement suivant. Soit $\mathcal{C}$ une chronique de profondeur $n+1$. Tous ses prédecesseurs sont de profondeur n et par monotonie de l'inclusion, il sont tous fréquents. En conséquence, chacun de ses prédécesseurs sera ajouté une et une seule fois à O et sera donc traité une et une seule fois par la boucle 4-21. Notons $\mathcal{C}'_1, \ldots, \mathcal{C}'_p$ ces prédécesseurs par ordre chronologique de traitement par la boucle 4-21 (si $i < j$ alors $\mathcal{C}'_i$ a été choisi par choisir (ligne 6) dans une itération antérieure à celle de $\mathcal{C}'_j$). Lors de l'itération où $\mathcal{C}'_1$ a été choisie par choisir, comme $\mathcal{C}'_1$ est fréquente, la procédure gén_succ lui sera appliquée, et $\mathcal{C}$ sera alors générée pour la première fois de l'exécution. Le test est_acceptable renverra vrai car :

1. $\mathcal{C}$ est fréquente et par monotonie de $\preceq$ il ne pourra pas y avoir de chroniques non-fréquentes plus générales dans nonF,

2. $\mathcal{C}$ ne sera pas dans `DéjàTraitées` puisqu'il s'agit de sa première génération.

$\mathcal{C}$ sera donc ajoutée à `O`. En revanche, pour les itérations des autres prédécesseurs $\mathcal{C}'_2$ à $\mathcal{C}'_p$, $\mathcal{C}$ sera générée mais pas ajoutée à `O` car présente dans `DéjàTraitées`. $\mathcal{C}$ sera donc ajoutée une et une seule fois à `O`.

Il est ensuite facile de montrer que si toute chronique fréquente est ajoutée à `O`, alors `F` contiendra finalement l'ensemble des chroniques minimales fréquentes.

La terminaison de l'algorithme est assurée par le raisonnement qui suit. Soit n_{max} la taille de la chronique la plus grande parmi toutes les chroniques fréquentes minimales de $\mathcal{S}$ construites à partir de $\mathcal{D}$. On est alors certain que `gén_succ`, qui ne s'applique qu'à des chroniques fréquentes, ne générera jamais de chroniques de taille $n_{max}+2$, car alors c'est qu'une chronique de taille $n_{max}+1$ serait fréquente. Soit $N_{\leq}(n_{max}+1)$ le nombre total de chroniques (fréquentes et non-fréquentes) construites à partir de $\mathcal{D}$. Comme le test `est_acceptable` assure qu'on ne mettra jamais deux fois la même chronique dans l'ensemble `O`, on est certain qu'à la i^e itération de la ligne 4, il reste $N_{\leq}(n_{max}+1)-i$ chroniques potentielles à traiter, donc que l'algorithme aura au pire encore $N_{\leq}(n_{max}+1)-i$ itérations à faire. Le paramètre $N_{\leq}(n_{max}+1)-i$ est donc un paramètre strictement décroissant de l'algorithme.

Au pire des cas, l'algorithme traitera donc $N_{\leq}(n_{max}+1)$ chroniques, où $N_{\leq}(n)$ est le nombre de chroniques de taille inférieure ou égale à n. En notant $N_{=}(n)$ le nom total de chroniques de taille n, on a $N_{\leq}(n_{max}+1) = N_{=}(2) + N_{=}(3) + \ldots + N_{=}(n_{max}+1)$. Si on suppose que $\mathcal{D}$ est constituée de graphes de contraintes étant tous de taille p, on a alors $N_{=}(n) = |\mathbb{E}|^n \times p^{C_n^2}$, et donc $N_{\leq}(n_{max}) = O(p^{n^2_{max}})$. Dans la pratique, l'introduction des tests de non-fréquence permet de ne pas traiter une branche de sous-chroniques dès lors qu'on sait qu'une chronique est non-fréquente. Pour la séquence $\mathcal{S}' = \langle (A,1)(B,3)(A,4)(C,4)(A,7)(B,8)\ (C,9)(B,10)(B,12)\rangle$ avec $f_{seuil}=2$, on a $N_{\leq} \simeq 10^7$ et pourtant la découverte peut nécessiter moins de 100 itérations selon le choix de l'heuristique (cf. section 4.2).

4.2 Amélioration des performances et de la pertinence par l'interactivité

L'algorithme exposé précédemment est conçu de telle sorte qu'il permette d'y introduire la possibilité de prendre en compte facilement des contraintes spécifiées par l'utilisateur. Chaque contrainte est une condition supplémentaire traitée par `est_acceptable` lors de l'ajout des successeurs. En procédant de la sorte, le processus de découverte restera complet pour toute contrainte monotone pour l'inclusion de chroniques telle la taille maximale de la chronique, le fait d'être plus général qu'une chronique spécifiée par l'utilisateur, etc. La contrainte d'inclusion dans une chronique est au contraire *antimonotone*. On peut par exemple imaginer que l'utilisateur ne s'intéresse qu'à des chroniques comprenant au moins les événements A et B telles que B succède à A, c'est-à-dire qu'il recherche toutes les chroniques $\mathcal{C}$ vérifiant $\mathcal{C} \preceq A[0,+\infty]B$. Pour prendre une telle contrainte en compte on pourra initialiser le processus de découverte à $\texttt{O} = A[0,+\infty]B$, réduisant ainsi considérablement l'espace de recherche.

Il est également immédiat d'introduire dans cet algorithme des heuristiques dans le choix effectué par `choisir` de la prochaine chronique à traiter. En effet, si `choisir` est capable de choisir parmi l'ensemble des chroniques à traiter les plus «intéressantes», alors l'ensemble `F` aura tendance à contenir plus rapidement les chroniques qui intéressent l'utilisateur. L'idée du choix de l'heuristique est que dans un temps très court, on puisse extraire une grande partie des chroniques les plus pertinentes même si le pourcentage d'exploration de l'espace des chroniques candidates est encore très faible. On pourra alors se satisfaire du résultat temporaire de l'extraction et couper l'exécution. Pour trouver de telles heuristiques, il est possible de s'inspirer des nombreuses mesures d'intérêt sur les motifs existant dans la littérature (Geng et Hamilton, 2006). Ces heuristiques sont des fonctions visant à mesurer l'intérêt d'un motif sous un autre angle que celui classique de la *fréquence* du motif. On trouve par exemple des mesures visant à spécifier le niveau d'*utilité* pour l'utilisateur, de *nouveauté* ou de *particularité* par rapport aux autres motifs. Ces mesures d'intérêt ont dans la grande majorité été conçues pour l'application à des motifs de type «règle d'association» et très peu à des motifs séquentiels et encore moins au cas particulier des chroniques. Il y a donc un travail important à effectuer d'innovation et d'adaptation de ces mesures pour les données séquentielles.

Le couplage des heuristiques et des contraintes peut mener à un cercle vertueux que nous cherchons à engendrer. En effet, nous imaginons un cycle de découverte interactive dans lequel les premiers résultats retournés par les premières itérations de l'algorithme permettent à l'utilisateur de spécifier des contraintes supplémentaires pour préciser sa recherche. De même, les premiers résultats de la deuxième exécution permettent de peaufiner les contraintes, et ainsi de suite. À chaque interaction avec l'utilisateur, des connaissances sur les chroniques recherchées par l'utilisateur seront ainsi capitalisées et peuvent être exploitées pour élaborer des heuristiques reflétant plus le point de vue de l'utilisateur.

Nous avons appliqué à la séquence $\mathcal{S}'$ trois heuristiques différentes et très simples : *choix aléatoire*, *FIFO* et *LIFO*. Pour *choix aléatoire*, l'algorithme proposé a nécessité 215 itérations et 81 comptages (ligne 10), pour *FIFO* 90442 et 223, et pour *LIFO* 74 et 53. Ces heuristiques, pourtant peu élaborées, mettent en évidence que comme attendu que le choix de l'heuristique a une grande influence sur l'exécution du processus.

5 Conclusion

L'algorithme présenté dans cette article permet l'extraction de motifs temporels à partir d'une séquence d'événements avec expression de contraintes temporelles numériques entre les événements du motifs. La méthode proposée est complète, contrairement à celle qui a été proposée par Duong (2001). La contre-partie de la complétude de la découverte et de la forte expressivité des motifs recherchés est la grande complexité en temps de la méthode. C'est pour cela que le processus de découverte proposé permet d'y introduire facilement des contraintes et des heuristiques permettant respectivement de réduire l'espace de recherche et d'orienter la recherche en priorité vers les chroniques les plus intéressantes pour l'utilisateur. L'autre avantage de l'utilisation d'heuristiques est qu'il est possible pour l'utilisateur, à la manière d'un algorithme anytime, de se satisfaire à un tout moment du processus des solutions partielles découvertes. L'hypothèse faite est que ces solutions partielles donnent assez d'indications à l'utilisateur quant à la pertinence de la requête qu'il a formulée et lui permet ainsi de reformuler une nouvelle requête plus contrainte. Ce processus interactif se poursuit jusqu'à ce que l'ensemble des contraintes accumulées au cours des itérations compensent la grande complexité et permettent de déboucher sur l'ensemble complet des chroniques satisfiant les contraintes en un temps raisonnable.

Ce qui est présenté ici est donc un «cadre de travail» pour la recherche interactive de chroniques, dont les travaux futurs doivent se porter d'une part sur la recherche de telles heuristiques et de contraintes, ainsi que sur l'étude de l'impact de ces heuristiques et contraintes sur l'efficacité du processus de découverte en termes de *rapidité d'exécution* et de *pertinence* des motifs découverts par rapport aux attentes de l'utilisateur.

Nous menons actuellement d'autres recherches sur l'extension du formalisme des chroniques à des événements persistants. Un tel événement a une date de début et une date de fin différentes, et il est à vrai dire plus exact de modéliser les *traces d'interaction* avec des événements persistants. Certains travaux comme ceux de Patel et al. (2008) ont rendu possible la découverte complète de motifs temporels dans des séquences d'événements persistants, mais ils ne permettent pas l'expression de contraintes numériques.

Références

Cram, D., B. Fuchs, Y. Prié, et A. Mille (2008). An approach to User-Centric Context-Aware Assistance based on Interaction Traces. In *MRC2008 : fifth International Workshop on Modeling and Reasoning in Context*.

Dousson, C., P. Gaborit, et M. Ghallab (1993). Situation recognition : Representation and algorithms. In *IJCAI*, pp. 166–174.

Duong, M. T. V. (2001). *Découverte de chroniques à partir de journaux d'alarmes. Application à la supervision de réseaux de télécommunications*. Ph. D. thesis, Institut National Polytechnique de Toulouse.

Fauré, C. (2007). *Découverte de réseau pertinents par l'implémentation d'un réseau bayésien : application à l'industrie aéronautique*. Ph. D. thesis, INSA de Lyon.

Geng, L. et H. J. Hamilton (2006). Interestingness measures for data mining : A survey. *ACM Comput. Surv. 38*(3), 9.

Lin, M.-Y. et S.-Y. Lee (2004). Interactive sequence discovery by incremental mining. *Inf. Sci. Inf. Comput. Sci. 165*(3-4), 187–205.

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.

Parthasarathy, S., M. J. Zaki, M. Ogihara, et S. Dwarkadas (1999). Incremental and interactive sequence mining. In *CIKM*, pp. 251–258.

Patel, D., W. Hsu, et M. L. Lee (2008). Mining relationships among interval-based events for classification. In *SIGMOD '08 : Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, New York, NY, USA, pp. 393–404. ACM.

Summary

This papers proposes a complete method for the discovery of frequent temporal patterns with numerical constraints from a sequence of events. These patterns are called *chronicles*. *Chronicles* are very expressive compared to more "common" sequential patterns like serial and parallel episodes. Consequently, the complete discovery process has a high complexity. We propose an algorithm that solves the complete dicovery problem by a heuristic search and enabling user constraints. It is also discussed the opportunity of taking advantage from user interactions to make this very complex process more efficient and to have an acceptable execution time.