Une contrainte globale pour l'extraction de motifs séquentiels

Amina Kemmar[1] **, Yahia Lebbah**[1] **, Samir Loudni**[2] **,**
Patrice Boizumault[2] **, Thierry Charnois**[3]

1. *Université d'Oran 1, Lab. LITIO, B.P. 1524 El-M'Naouar, 31000 Oran, Algérie*
   kemmar.amina@edu.univ-oran1.dz,lebbah.yahia@univ-oran.dz
2. *Université de Caen Basse-Normandie, GREYC, 14032 Caen, France*
   samir.loudni@unicaen.fr,patrice.boizumault@unicaen.fr
3. *Université Paris-Nord, LIPN, 93430 Villetaneuse, France*
   thierry.charnois@lipn.univ-paris13.fr

RÉSUMÉ. *L'extraction de motifs séquentiels sous contraintes est un problème majeur en fouille de données. De nombreuses méthodes ad hoc efficaces on été proposées pour le résoudre, mais elles souffrent toutes d'un manque de généricité. Des approches basées sur la Programmation Par Contraintes (PPC) ont été récemment proposées, mais elles ne sont pas encore assez efficaces, principalement en raison de l'encodage utilisé. Dans cet article, nous proposons une contrainte globale basée sur le principe de projection qui permet de remédier à cet inconvénient. Les expérimentations menées montrent que notre approche surpasse clairement les approches PPC, et demeure compétitive avec les méthodes ad hoc sur de grandes bases de séquences.*

ABSTRACT. *Sequential pattern mining under constraints is a challenging data mining task. Many efficient ad hoc methods have been developed for mining sequential patterns, but they are all suffering from a lack of genericity. Recent works have investigated Constraint Programming (CP) methods, but they are not still effective because of their encoding. In this paper, we propose a global constraint based on the projected databases principle which remedies to this drawback. Experiments show that our approach clearly outperforms CP approaches and competes well with ad hoc methods on large datasets.*

MOTS-CLÉS : *fouille de motifs séquentiels, projection préfixée, programmation par contraintes, contrainte globale.*

KEYWORDS: *sequential pattern mining, prefix-projection, constraint programming, global constraint.*

DOI:10.3166/RIA.30.675-703

1. Introduction

L'extraction de motifs séquentiels (Agrawal, Srikant, 1995) est l'une des tâches les plus étudiées en fouille de données. Elle possède de nombreuses applications dans des domaines tels que la bioinformatique, l'analyse de weblogs, la détection d'intrusions, les réseaux de télécommunications, et la fouille de textes. L'extraction de motifs séquentiels a pour but la découverte de régularités dans des données se présentant sous forme de séquences. L'extraction de motifs sous contraintes contribue à réduire le nombre de motifs en ciblant les motifs potentiellement intéressants (Dong, Pei, 2007). L'usage des contraintes permet aussi de concevoir des algorithmes plus efficaces en réduisant l'espace de recherche.

De nombreuses méthodes ont été proposées pour l'extraction de *motifs séquentiels* (e.g. `GSP` (Srikant, Agrawal, 1996), `SPADE` (Zaki, 2001), `PrefixSpan` (Pei *et al.*, 2001)), ou de *motifs fermés* (e.g. `CloSpan` (Yan *et al.*, 2003), `BIDE` (Wang, Han, 2004)), ou encore de motifs séquentiels satisfaisant une certaine *expression régulière* (e.g. `SPIRIT` (Garofalakis *et al.*, 2002), `SMA` (Trasarti *et al.*, 2008)). Une étude comparative sur les différentes classes de contraintes pouvant être prises en compte pour la découverte de motifs utiles (e.g., contrainte d'expression régulière, contrainte de taille, contrainte d'appartenance d'items, contrainte d'agrégat) est présentée de manière détaillée dans (Dong, Pei, 2007).

Mais, si ces méthodes sont efficaces, elles ne peuvent traiter que quelques classes particulières de contraintes (*monotones*, *anti-monotones*, *préfixe-monotones* ou *préfixe-anti-monotones*). Ces méthodes souffrent toutes de manque de généricité pour traiter et pousser simultanément plusieurs combinaisons de classes de contraintes. Ce manque de généricité est un frein à la découverte de motifs pertinents car la prise en compte de nouvelles contraintes ou leur combinaison avec d'autres types de contraintes nécessite la conception et le développement de méthodes spécialisées.

Pour lever ce frein, des travaux récents proposent de modéliser la fouille de séquences à l'aide de la Programmation Par Contraintes (PPC) (Coquery *et al.*, 2012 ; Métivier *et al.*, 2013 ; Kemmar *et al.*, 2014 ; Négrevergne, Guns, 2015). Le point commun de ces travaux est de modéliser le problème de la fouille de séquences en un Problème de Satisfaction de Contraintes (CSP). Une telle modélisation présente l'avantage d'être flexible en permettant de définir de nouvelles contraintes sans s'occuper de leur résolution. Cependant, un obstacle essentiel de ces approches déclaratives est qu'elles ne sont pas encore suffisamment compétitives car elles ne passent pas à l'échelle comparées aux méthodes spécialisées. En effet, l'encodage PPC de la base de séquences, qui utilise des contraintes réifiées, constitue une forte limitation sur la taille des bases qui peuvent être traitées. Par ailleurs, l'encodage PPC du motif recherché rend difficile la prise en compte de l'(anti)-monotonie de la contrainte de fréquence lors de la phase d'énumération des motifs candidats. Par conséquent, la conception de nouveaux modèles déclaratifs efficaces pour l'extraction de motifs pertinents dans une base de séquences est un défi majeur pour la PPC.

Nous explorons dans cet article une autre voie consistant à introduire les techniques utilisées en fouille de données pour améliorer les performances de la PPC pour la fouille de séquences.

Premièrement, nous proposons la contrainte globale PREFIX-PROJECTION pour l'extraction de motifs séquentiels. PREFIX-PROJECTION utilise un encodage concis et son filtrage exploite le principe de *la projection préfixée* (Pei *et al*., 2001). L'idée principale consiste à diviser la base initiale de séquences en plusieurs bases plus petites (*bases projetées*) déduites à partir des préfixes fréquents identifiés jusqu'alors. Ensuite, dans chaque base projetée, il faut étendre le préfixe courant avec les items fréquents dans cette base projetée. Deuxièmement, nous montrons comment notre encodage permet de mettre en œuvre différents types de contraintes (appartenance des items, taille des items et expressions régulières) et de les combiner simultanément. Enfin, les résultats expérimentaux montrent que notre approche surpasse clairement les approches PPC existantes et concurrence les méthodes spécialisées sur de grandes bases de séquences. Il est à noter que les expérimentations montrent que notre approche permet de passer à l'échelle, alors qu'il s'agit d'un enjeu majeur pour les approches PPC existantes.

Par rapport au papier de conférence (Kemmar *et al*., 2015), les deux points cidessous ont été améliorés de manière significative :

– La contrainte globale PREFIX-PROJECTION est présentée de manière plus approfondie. Chaque définition ou proposition est illustrée à travers un exemple. Les comportements de l'algorithme de filtrage et la recherche des solutions sont également analysés.

– Nous donnons plus d'explications sur le principe des méthodes existantes. En particulier, nous illustrons une comparaison entre notre approche PREFIX-PROJECTION et l'approche PPC proposée dans (Négrevergne, Guns, 2015) (cf. section 4.6).

L'article est organisé comme suit. La section 2 rappelle les préliminaires. La section 3 donne un aperçu critique des méthodes spécialisées et des approches PPC existantes. La section 4 présente la contrainte globale PREFIX-PROJECTION. La section 5 présente les expérimentations que nous avons effectuées. Enfin, nous concluons et dressons quelques perspectives.

2. Préliminaires

Cette section introduit les définitions usuelles utilisées respectivement en fouille de séquences et en programmation par contraintes.

2.1. Extraction de motifs séquentiels

Soit $\mathcal{I}$ un ensemble fini de n éléments (*items*). Le langage de séquences non vides correspond à $\mathcal{L}_{\mathcal{I}} = \mathcal{I}^+$.

Tableau 1. SDB_1 : Un exemple de base de séquences

sid	Séquence
1	$\langle ABCBC \rangle$
2	$\langle BABC \rangle$
3	$\langle AB \rangle$
4	$\langle BCD \rangle$

DÉFINITION 1 (séquence, base de séquences). — *Une séquence s définie sur $\mathcal{L}_{\mathcal{I}}$ est une liste ordonnée d'éléments $\langle s_1 s_2 \dots s_m \rangle$, où $s_i \in \mathcal{I}$ est un item, $1 \leq i \leq m$. m est appelé la longueur de la séquence s. Une base de séquences SDB est un ensemble de tuples (sid, s), où s est la séquence et sid représente son identifiant.*

EXEMPLE 2. — Soit la base de séquences SDB_1 du tableau 1. SDB_1 contient quatre séquences où l'ensemble des items $\mathcal{I} = \{A, B, C, D\}$. La séquence $s = \langle ABCBC \rangle$ est composée de 5 items. On remarque que les items B et C apparaissent deux fois dans la séquence s. La longueur de s est 5, donc on dit que s est une 5-séquence. □

DÉFINITION 3 (sous-séquence, la relation $\preceq$). — *Une séquence $s = \langle s_1 \dots s_t \rangle$ est une sous-séquence d'une séquence $s' = \langle s'_1 \dots s'_n \rangle$ (ou s' est une sur-séquence de s), notée ($s \preceq s'$), ssi $t \leq n$ et il existe des entiers $1 \leq j_1 \leq \dots \leq j_t \leq n$ tels que $s_i = s'_{j_i}$ pour tout $1 \leq i \leq t$. On dit aussi que s est contenue dans s'. Un tuple (sid', s') contient une séquence s si $s \preceq s'$.*

DÉFINITION 4 (couverture, support). — *Soient SDB une base de séquences et s une séquence. La couverture de s dans SDB est l'ensemble de tous les tuples de SDB qui contiennent s : $cover_{SDB}(s)$=$\{(sid, s') \in SDB \mid s \preceq s'\}$, et son support est défini par $sup_{SDB}(s) = |cover_{SDB}(s)|$.*

DÉFINITION 5 (motif séquentiel). — *Soit $minsup$ un seuil de support minimal. On dit qu'une séquence s est fréquente dans une base SDB, si son support contient au moins $minsup$ éléments : $sup_{SDB}(s) \geq minsup$. s est appelée motif séquentiel (Agrawal, Srikant, 1995).*

EXEMPLE 6. — Considérons la base de séquences du tableau 1. Soit la séquence $s = \langle AC \rangle$, alors $cover_{SDB1}(s) = \{(1, s_1), (2, s_2)\}$. Si on considère $minsup = 2$, alors $s = \langle AC \rangle$ est un motif séquentiel car $sup_{SDB_1}(s) \geq 2$. □

DÉFINITION 7 (Extraction de motifs séquentiels (EMS)). — *Etant donnés un seuil de support minimal $minsup \geq 1$ et une base de séquences SDB, le problème d'extraction des motifs séquentiels consiste à trouver toutes les séquences s telles que $sup_{SDB}(s) \geq minsup$.*

EXEMPLE 8. — La résolution du problème EMS appliqué sur la base de séquences SDB_1 avec $minsup = 3$ consiste à trouver l'ensemble des motifs qui apparaissent au moins dans 3 séquences de SDB_1. Nous obtenons ainsi les motifs séquentiels suivants : $\langle A \rangle$, $\langle B \rangle$, $\langle C \rangle$, $\langle AB \rangle$ et $\langle BC \rangle$. □

2.2. Projection préfixée et bases projetées

Nous reprenons ici les définitions nécessaires pour introduire le principe de la *projection préfixée* (Pei *et al*., 2001).

DÉFINITION 9 (préfixe, projection, suffixe). — *Soient* $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ *et* $\beta = \langle \beta_1 \dots \beta_n \rangle$ *deux séquences* (*avec* $m \leq n$).

– *La séquence* α *est un préfixe de* β *ssi* $\forall i \in [1..m], \alpha_i = \beta_i$.

– *La séquence* $\beta = \langle \beta_1 \dots \beta_n \rangle$ *est une projection de la séquence* s *par rapport à* α*, ssi (1)* $\beta \preceq s$*, (2)* α *est un préfixe de* β *et (3) il n'existe pas de sur-séquence propre* β' *de* β *telle que* $\beta' \preceq s$ *et* α *est un préfixe de* β'.

– *La séquence* $\gamma = \langle \beta_{m+1} \dots \beta_n \rangle$ *est appelée suffixe de* s *par rapport au préfixe* α*. Ainsi, nous avons* $\beta = concat(\alpha, \gamma)$*, avec "concat" est l'opérateur standard de concaténation.*

DÉFINITION 10 (Base de séquences projetées). — *Soit* SDB *une base de séquences. La base projetée (ou* α*-projection) de* SDB *par rapport à* α*, notée* $SDB|_\alpha$*, est l'ensemble de tous les suffixes des projections des séquences de* SDB *par rapport à* α.

EXEMPLE 11. — Soient la base de séquences SDB_1 du tableau 1, et les trois préfixes $\langle A \rangle$, $\langle AB \rangle$ et $\langle ABC \rangle$. Nous avons :

– $SDB_1|_{<A>}$ = $\{(1, \langle BCBC \rangle), (2, \langle BC \rangle), (3, \langle B \rangle)\}$.

– $SDB_1|_{<AB>}$ = $\{(1, \langle CBC \rangle), (2, \langle C \rangle), (3, \langle \rangle)\}$.

– $SDB_1|_{<ABC>}$ = $\{(1, \langle BC \rangle), (2, \langle \rangle)\}$.

□

(Pei *et al*., 2001) ont proposé un algorithme efficace, appelé `PrefixSpan`, pour l'extraction de motifs séquentiels basé sur le principe de la *projection préfixée*. À chaque étape, cet algorithme construit récursivement une nouvelle base de séquences de taille inférieure (base projetée). Ainsi, après avoir extrait tous les motifs fréquents de taille 1, c'est-à-dire contenant un seul item, une base projetée de SDB est calculée pour chaque item. Cette base projetée contient les suffixes des séquences ayant pour préfixe le motif fréquent de taille 1 contenant l'item. L'étape suivante calcule les motifs fréquents de taille 2 à partir des bases projetées obtenues à l'étape 1 ainsi que les bases projetées associées à ces nouveaux motifs. Le processus se répète, la taille des bases projetées se réduisant à chaque étape.

EXEMPLE 12. — Prenons l'exemple de la base SDB_1 du tableau 1. Supposons que $minsup = 2$. `PrefixSpan` commence par trouver les items fréquents. Pour cela, une passe sur la base SDB_1 va permettre de collecter le nombre de séquences supportant chaque item rencontré et donc d'évaluer le support des items de la base. Les items trouvés sont (sous la forme <item>:support) : $\langle A \rangle : 3$, $\langle B \rangle : 4$ et $\langle C \rangle : 3$. Ensuite, les séquences de SDB_1 sont projetées en trois sous-ensembles disjoints, puisqu'il y a trois préfixes de taille 1 (i.e. les trois items fréquents). Ces sous-ensembles seront : (1) les motifs séquentiels ayant pour préfixe $\langle A \rangle$, (2) ceux ayant pour préfixe $\langle B \rangle$ et (3) ceux ayant pour préfixe $\langle C \rangle$. Par exemple, la base projetée de SDB_1 selon le pré-

fixe $\langle A\rangle$, notée $SDB_1|_{\langle A\rangle}$, est composée de 3 suffixes : $\{(1, \langle BCBC\rangle), (2, \langle BC\rangle), (3, \langle B\rangle)\}$. Une passe sur $SDB_1|_{<A>}$ permet alors d'obtenir les motifs séquentiels de longueur 2 ayant $\langle A\rangle$ pour préfixe commun : $\langle AB\rangle : 3$ et $\langle AC\rangle : 2$. Ainsi, de façon récursive, $SDB_1|_{<A>}$ peut être partitionnée en deux sous-ensembles : (1) les motifs séquentiels ayant pour préfixe $\langle AB\rangle$ et (2) ceux ayant pour préfixe $\langle AC\rangle$. Ces motifs peuvent alors former de nouvelles bases projetées, et chacune de ces bases peut être utilisée en entrée de l'algorithme, toujours de manière récursive. Ce processus de α-projection des bases projetées se termine lorsque aucun super-motif séquentiel ne peut être obtenu. □

La proposition 13 établit une propriété pour calculer le support d'une séquence γ dans $SDB|_\alpha$ (Pei *et al.*, 2001).

PROPOSITION 13 (Calcul de support). — *Pour toute séquence β ayant pour préfixe α et pour suffixe γ, avec $\beta = concat(\alpha, \gamma)$, $sup_{SDB}(\beta) = sup_{SDB|_\alpha}(\gamma)$.*

Cette proposition garantit que seules les séquences dans SDB obtenues par extension du motif α seront considérées pour le calcul du support de la séquence γ. En outre, seuls les suffixes de $SDB|_\alpha$ doivent être considérés pour le calcul de ce support.

EXEMPLE 14. — Considérons la séquence $\beta = \langle ABC\rangle$. Supposons que $\alpha = \langle AB\rangle$, selon la proposition 13, $sup_{SDB_1}(\beta) = sup_{SDB_1}(concat(\langle AB\rangle, \langle C\rangle)) = sup_{SDB_1|_{\langle AB\rangle}}(\langle C\rangle)$. Nous avons $SDB_1|_{\langle AB\rangle} = \{(1, \langle CBC\rangle), (2, \langle C\rangle), (3, \langle\rangle)\}$. Puisque l'item C apparaît seulement dans la première et la deuxième séquence dans $SDB|_{\langle AB\rangle}$, alors $sup_{SDB_1}(\langle ABC\rangle) = sup_{SDB_1|_{\langle AB\rangle}}(\langle C\rangle) = 2$. □

2.3. Extraction de motifs séquentiels sous-contraintes

Dans la pratique, le nombre de motifs séquentiels fréquents peut être important. Une façon de réduire leur nombre est d'utiliser des contraintes. Les contraintes permettent à l'utilisateur de définir plus précisément les motifs qu'il considère pertinents (Pei *et al.*, 2002). Cette section présente les contraintes les plus utilisées en fouille de séquences (Pei *et al.*, 2002). D'autres contraintes (Zaki, 2000), qui ne sont pas abordées dans ce travail, peuvent être aussi intéressantes pour l'utilisateur telles que la contrainte d'inclusion, d'agrégat ou contrainte de gap. Cette dernière permet d'imposer des restrictions sur la distance qui sépare deux items consécutifs. Son implémentation nécessite des modifications au sein de l'algorithme de filtrage (Pour plus de détails, voir (Kemmar *et al.*, 2016)).

1. **Contrainte de fréquence.** Cette contrainte, que nous avons déjà introduite, permet de restreindre l'ensemble des motifs extraits aux motifs fréquents selon un seuil de fréquence minimal $minsup$.

2. **Contrainte de longueur minimale.** Cette contrainte, notée $minSize(p, \ell_{min})$, spécifie la taille minimale (en nombre d'items) des motifs extraits (i.e. les motifs extraits doivent contenir au moins ℓ_{min} items).

EXEMPLE 15. — Reprenons l'exemple 6 en imposant la contrainte $minSize(p, 3) \wedge sup_{SDB_1}(p) \geq 2$. Seuls deux motifs sont extraits $p_1 = \langle ABC \rangle$ et $p_2 = \langle BBC \rangle$. □

3. **Contrainte de longueur maximale.** Cette contrainte, notée $maxSize(p, \ell_{max})$, spécifie la taille maximale (en nombre d'items) des motifs extraits (i.e. les motifs extraits doivent contenir au plus ℓ_{max} items).

EXEMPLE 16. — Par exemple, en imposant la contrainte $maxSize(p, 2) \wedge sup_{SDB_1}(p) \geq 3$, seuls les motifs suivants sont extraits : $\langle A \rangle$, $\langle B \rangle$, $\langle C \rangle$, $\langle AB \rangle$ et $\langle BC \rangle$. □

4. **Contrainte d'item.** Cette contrainte spécifie le sous-ensemble d'items qui doivent apparaître dans les motifs extraits : $item(p, t) \equiv (\exists i \in 1..|p|, p_i = t)$.

EXEMPLE 17. — Considérons les trois contraintes suivantes : $sup_{SDB_1}(p) \geq 3 \wedge maxSize(p, 2) \wedge item(p, C)$, définies sur les séquences du tableau 1. Seuls deux motifs sont extraits $p_3 = \langle C \rangle$ et $p_4 = \langle BC \rangle$. □

5. **Contrainte d'expression régulière.** La contrainte d'expression régulière, notée $reg(p, exp)$, assure que le motif p doit être reconnu par un automate d'états finis associé à l'expression régulière exp (Garofalakis *et al.*, 2002).

EXEMPLE 18. — Considérons la contrainte d'expression régulière $reg(p, exp)$, où $exp = B\{BC|D\}$. Le motif $p_5 = \langle BBC \rangle$ est extrait de SDB_1. □

2.4. CSP et Contraintes globales

Cette section rappelle les définitions nécessaires à la présentation de notre approche.

Programmation par contraintes (PPC). La PPC est un paradigme puissant pour résoudre des problèmes combinatoires modélisés sous forme de contraintes, se basant sur des travaux issus de l'intelligence artificielle et de la recherche opérationnelle. La PPC se base sur le principe suivant : (1) l'utilisateur spécifie le problème d'une façon déclarative comme un *problème de satisfaction de contraintes* (CSP) ; (2) le solveur cherche l'ensemble complet et correct des solutions du problème. De cette manière, la spécification du problème est séparée de la stratégie de recherche.

Problème de satisfaction de contraintes (CSP). Un CSP est un triplet $(X, \mathcal{D}, \mathcal{C})$ où $X = \{X_1, \ldots, X_n\}$ est un ensemble fini de variables ayant pour domaines finis $\mathcal{D} = \{D_1, \ldots, D_n\}$ et $\mathcal{C} = \{C_1, \ldots, C_m\}$ est un ensemble de contraintes où chaque C_i est une condition sur un sous-ensemble de X. On désignera par portée de la contrainte C_i, les variables impliquées dans C_i. Une instanciation σ est une application des variables de X vers leurs domaines : $\forall X_i \in X, \sigma(X_i) \in D(X_i)$. L'objectif est de trouver une instanciation $\sigma(X_i) = d_i$ ($X_i = d_i$) avec $d_i \in D_i$ pour chaque variable $X_i \in X$, telle que toutes les contraintes soient satisfaites. Une telle instanciation est appelée *solution* pour le CSP.

Filtrage. En PPC, le processus de résolution consiste à combiner itérativement des phases de *recherche* et de *propagation*. La phase de recherche consiste à énumérer toutes les instanciations partielles possibles des variables jusqu'à trouver une solu-

tion ou prouver qu'il n'en existe pas. L'espace de recherche peut être représenté par un arbre : à chaque nœud de l'arbre correspond une variable et à chaque branche une des différentes instanciations possibles des variables. La phase de *propagation de contraintes* permet de réduire l'espace de recherche en *filtrant*, à chaque nœud de l'arbre, les valeurs des domaines des variables qui ne peuvent participer à aucune solution, en maintenant une propriété de cohérence locale sur les contraintes individuelles telle que la *cohérence de domaine*.

DÉFINITION 19 (Cohérence de domaine). — *Une contrainte C définie sur X vérifie la cohérence de domaine, si et seulement si, pour chaque X_i de la portée de C et $d_i \in D(X_i)$, il existe une instanciation σ qui satisfait C telle que $\sigma(X_i) = d_i$. Cette instanciation est appelée support.*

Ainsi, à chaque contrainte est associé un *propagateur* (i.e. un algorithme de filtrage) permettant de supprimer toutes les valeurs des domaines des variables de sa portée qui ne satisfont pas la contrainte. Comme les variables peuvent participer dans plusieurs contraintes, les domaines ainsi mis à jour sont propagés aux autres contraintes, en activant les propagateurs de ces contraintes. Ce processus de propagation de contraintes est répété sur toutes les contraintes jusqu'à ce qu'aucune suppression ne soit possible ou qu'un domaine devienne vide.

Les contraintes globales décrivent une propriété que doit satisfaire un ensemble de variables. Les contraintes globales sont un atout important pour la modélisation en PPC. De plus, en exploitant la structure de la contrainte, elles permettent la mise en œuvre de tests de cohérence et d'algorithmes de filtrage très efficaces en temps et peu coûteux en espace. De nos jours, les contraintes globales sont considérées comme l'un des plus importants composants d'un solveur de contraintes. Nous présentons brièvement deux contraintes globales, `Among` et `Regular`, utilisées pour modéliser les différentes contraintes décrites dans la section 2.3.

1. *Contrainte* `Among`. Cette contrainte restreint le nombre d'occurrences de certaines valeurs dans une séquence de n variables (Beldiceanu, Contejean, 1994).

DÉFINITION 20 (Beldiceanu, Contejean, 1994). — *Soient $X = \langle X_1, X_2, ..., X_n \rangle$ une séquence de n variables, V un ensemble de valeurs et D^X le produit cartésien des domaines des variables de X. Soient l et u deux entiers tel que $0 \leq l \leq u \leq n$, la contrainte $\texttt{Among}(X, V, l, u)$ impose que le nombre de variables instanciées à des valeurs dans V, soit compris entre l et u :*

$$\texttt{Among}(X, V, l, u) = \{\sigma \in D^X \mid l \leq \mid \{i, \sigma(X_i) \in V\} \mid \leq u\}$$

2. *Contrainte* `Regular`. Soient M un automate fini déterministe et $X = \langle X_1, X_2, ..., X_n \rangle$ une séquence de n variables. La contrainte $\texttt{Regular}(X, M)$ impose que X appartienne au langage régulier reconnu par M.

DÉFINITION 21 (Pesant, 2004). — *Soient M un automate fini déterministe, $\mathcal{L}(M)$ le langage régulier reconnu par M et $X = \langle X_1, X_2, ..., X_n \rangle$ une séquence de n variables :*

$$\texttt{Regular}(X, M) = \{\sigma \in D^X \mid \sigma \in \mathcal{L}(M)\}$$

3. Etat de l'art

Dans cette section, nous passons en revue les différentes méthodes proposées pour l'extraction de motifs séquentiels. Nous commençons par les méthodes spécialisées, puis nous terminons par les méthodes basées PPC.

3.1. Méthodes spécialisées pour le problème EMS

GSP (Srikant, Agrawal, 1996) est le premier algorithme proposé pour l'extraction de motifs séquentiels. Il se base sur l'approche générer et tester. Plus tard, deux grandes catégories de méthodes ont été proposées :

1. Les algorithmes de recherche en profondeur basés sur un format vertical pour la représentation de la base de séquences. Dans ce format, on associe pour chaque k-séquence (une séquence de longueur k) l'ensemble des couples (sid, eid) qui représente respectivement l'identifiant de la séquence dans la base qui l'inclus et la position de départ de chacune de ses occurrences. e.g. cSpade qui intègre les contraintes gap, item et longueur (Zaki, 2000), SPADE (Zaki, 2001) ou SPAM (Ayres *et al*., 2002).

2. Les algorithmes basés sur l'extension de motifs en exploitant le principe de la projection préfixée tels que PrefixSpan (Pei *et al*., 2001) et ses extensions, e.g. CloSpan pour l'extraction de motifs fermés (Yan *et al*., 2003) ou Gap-BIDE (Li *et al*., 2012) qui traite la contrainte de gap.

Dans (Garofalakis *et al*., 2002), les auteurs ont proposé SPIRIT, un algorithme qui se base sur GSP pour l'extraction de motifs séquentiels sous la contrainte d'expressions régulières. Plus tard, (Trasarti *et al*., 2008) ont introduit SMA, une nouvelle approche basée sur un type spécialisé de réseaux de Petri qui permet de produire pour chaque séquence dans la base uniquement les sous-séquences satisfaisant l'ER donnée en entrée. Deux variantes de SMA ont été proposées : SMA-1P (SMA one pass) et SMA-FC (SMA Full Check). SMA-1P parcourt les séquences une par une, en stockant tous les motifs valides dans une table de hachage pour le calcul de leurs supports ; alors que SMA-FC permet un élagage des motifs selon la fréquence durant la recherche. D'autres contraintes telles que les expressions régulières, la longueur ou les agrégats sont détaillées dans (Pei *et al*., 2002).

Mais, si ces méthodes sont efficaces, elles ne peuvent traiter que quelques classes particulières de contraintes (*monotones*, *anti-monotones*, *préfixe-monotones* ou *préfixe-anti-monotones*). Ces méthodes souffrent toutes de manque de généricité pour traiter et pousser simultanément plusieurs combinaisons de classes de contraintes. Ce manque de généricité est un frein à la découverte de motifs pertinents car la prise en compte de nouvelles contraintes ou leur combinaison avec d'autres types de contraintes nécessite la conception et le développement de méthodes spécialisées.

3.2. Méthodes PPC pour le problème EMS

Suite aux travaux récents de (Guns *et al*., 2011) sur la fertilisation croisée entre la fouille de motifs ensemblistes et la programmation par contraintes, plusieurs méthodes ont été proposées pour l'extraction de motifs séquentiels en utilisant la PPC.

(Coquery *et al*., 2012) ont proposé le premier modèle SAT pour la découverte de motifs avec wildcards dans une seule séquence, en considérant différents types de contraintes (e.g. fréquence, maximalité, fermeture). Un wildcard est un symbole spécial représenté par un vide, qui peut remplacer n'importe quel item de $\mathcal{I}$ dans une séquence. En s'appuyant sur la notion de motif séquentiel avec wildcards, (Kemmar *et al*., 2014) ont proposé un premier modèle CSP pour l'extraction de motifs séquentiels avec wildcards explicites dans une base de séquences. Ils montrent comment modéliser de nombreuses contraintes définies sur les motifs à extraire. Ces contraintes portent sur deux catégories de motifs : des contraintes définies sur des motifs locaux (e.g. fréquence, longueur, expression régulière, gap), ou des contraintes définies sur des ensembles de motifs comme les top-k motifs ou encore les sous-groupes pertinents. Dans ce modèle, les motifs séquentiels avec items non-contigus [1] sont modélisés en utilisant les wildcards comme des jokers. Si l'on considère la base de séquences du tableau 1, les deux motifs $\langle A\square C\rangle$ et $\langle A\square\square C\rangle$ sont considérés comme deux motifs différents. Par conséquent, la fouille de séquences avec wildcard ne permet pas de générer l'ensemble complet des motifs fréquents puisque certains motifs seront considérés comme non-fréquents. (Métivier *et al*., 2013) ont proposé le premier modèle CSP pour l'extraction de motifs séquentiels sans wildcards dans une base de séquences. Chaque séquence est encodée par un automate d'état fini qui capture toutes les sous-séquences de la séquence en question. Mais un tel encodage est très coûteux, ce qui rend cette approche peu efficace lorsque la taille de la base augmente. Récemment, dans (Négrevergne, Guns, 2015), les auteurs ont proposé deux encodages PPC pour le problème EMS. Le premier utilise une contrainte globale pour modéliser la relation de sous-séquence, alors que le deuxième encode d'une façon explicite cette relation en utilisant des variables et des contraintes additionnelles.

Toutes ces propositions encodent le problème EMS à l'aide de **contraintes réifiées**. Une contrainte réifiée permet d'associer une variable booléenne à une contrainte. Cette variable booléenne traduit la satisfaction de la contrainte : 1 si la contrainte est satisfaite, 0 sinon. Pour chaque séquence s de SDB, une contrainte réifiée est définie indiquant si le motif inconnu à extraire p est une sous-séquence (ou non) de s :

$$(S_s = 1) \Leftrightarrow (p \preceq s)$$

Cet encodage permet d'exprimer simplement la contrainte de fréquence : $freq(p) = \sum_{s \in SDB} S_s$. Mais, il présente un inconvénient majeur car il nécessite $|SDB|{=}m$ contraintes réifiées pour encoder toute la base de séquences. Ceci constitue une limitation forte sur la taille des bases qui peuvent êtres traitées.

1. On dit qu'un motif n'est pas contigu s'il contient au moins un wildcard avant son dernier item.

La plupart des propositions encodent la **relation de sous-séquence** ($p \preceq s$) en utilisant un ensemble de variables $Pos_{s,j}$ ($s \in SDB$ et $1 \leq j \leq \ell$, où ℓ est la longueur du motif p à encoder) pour déterminer la position où p apparaît dans s. Un tel encodage nécessite un grand nombre de variables intermédiaires ($m \times \ell$), ce qui rend ce processus plus coûteux en termes de temps de calcul. Afin de remédier à cet inconvénient, (Négrevergne, Guns, 2015) ont proposé une contrainte globale `exists-embedding` pour encoder la relation de sous-séquence et ont utilisé le principe de la fréquence projetée pour ne garder que les items localement fréquents. Pour cela, ils introduisent pour chaque séquence une variable auxiliaire utilisée pour conserver les éléments qui apparaissent après le préfixe courant. Pour éviter la recherche sur les éléments non-fréquents, une procédure de recherche spécifique (basée sur les domaines des variables auxiliaires) est utilisée, ce qui rend l'intégration assez complexe (voir la section 4.6 pour plus de détails). Mais encore une fois, cet encodage repose sur des contraintes réifiées et requiert m contraintes globales `exists-embedding`.

Pour pallier tous ces inconvénients, nous proposons dans la section suivante la contrainte globale PREFIX-PROJECTION qui exploite le principe de la projection préfixée afin d'encoder à la fois la relation de sous-séquence et la contrainte de fréquence. PREFIX-PROJECTION ne requiert ni contraintes réifiées ni variables supplémentaires pour encoder la relation de sous-séquence. Par conséquent, les contraintes usuelles (cf. section 2.3) définies sur les motifs peuvent être encodées directement en utilisant les contraintes (globales) du solveur de contraintes.

4. La contrainte globale PREFIX-PROJECTION

Tout d'abord, la section 4.1 décrit notre encodage des motifs à extraire. La section 4.2 définit formellement notre contrainte globale PREFIX-PROJECTION, ainsi que la cohérence maintenue. La section 4.3 montre comment la contrainte globale PREFIX-PROJECTION tire parti de l'anti-monotonie de la contrainte de support minimal pour réduire les domaines des variables. La section 4.4 détaille la construction des bases de séquences projetées. La section 4.5 s'intéresse à l'algorithme de filtrage de la contrainte PREFIX-PROJECTION et en évalue la complexité temporelle et spatiale. Enfin, la section 4.7 montre comment modéliser les contraintes décrites dans la section 2.3.

4.1. Encodage des motifs séquentiels

Soit P un motif inconnu de longueur ℓ. Le symbole $\square$ représente l'absence d'item et indique la fin de la séquence. Tout motif inconnu P est encodé par une séquence de ℓ variables $\langle P_1, P_2, \ldots, P_\ell \rangle$ t.q. $\forall i \in [1 \ldots \ell], D(P_i) = \mathcal{I} \cup \{\square\}$, avec les deux règles suivantes sur les domaines :

– Pour ignorer la séquence vide, le premier item de P doit être non vide : $\square \notin D_1$.

– Pour autoriser des motifs de taille inférieure à ℓ : $\forall i \in [1..(\ell-1)], (P_i = \square) \rightarrow (P_{i+1} = \square)$.

4.2. Définition et test de cohérence

La contrainte globale PREFIX-PROJECTION encode à la fois la relation de sous-séquence et la contrainte de fréquence minimale.

DÉFINITION 22 (Contrainte PREFIX-PROJECTION). — *Soient $P = \langle P_1, P_2, \ldots, P_\ell \rangle$ la séquence de variables représentant un motif inconnu de longueur ℓ et $minsup$ le seuil de support minimal. L'instanciation $\langle d_1, ..., d_\ell \rangle \in D(P_1) \times \ldots \times D(P_\ell)$ est une solution pour la contrainte* PREFIX-PROJECTION $(P, SDB, minsup)$ *si et seulement si $sup_{SDB}(\langle d_1, ..., d_\ell \rangle) \geq minsup$.*

PROPOSITION 23 (Test de cohérence de PREFIX-PROJECTION). — *La contrainte* PREFIX-PROJECTION $(P, SDB, minsup)$ *admet une solution ssi il existe une instanciation $\sigma = \langle d_1, ..., d_\ell \rangle$ des variables de P t.q. $SDB|_\sigma$ contient au moins $minsup$ suffixes des projections des séquences de SDB par rapport à σ, i.e. $|SDB|_\sigma| \geq minsup$.*

PREUVE. — C'est une conséquence directe de la proposition 13, car $sup_{SDB}(\sigma) = sup_{SDB|_\sigma}(\langle\rangle) = |SDB|_\sigma|$. Ainsi, les items fréquents dans les éléments de $SDB|_\sigma$ représentent des supports de σ pour la contrainte PREFIX-PROJECTION $(P, SDB, minsup)$, ce qui assure que $|SDB|_\sigma| \geq minsup$.

■

La proposition suivante permet de caractériser les valeurs du domaine de la variable non instanciée P_{i+1} (appelée aussi variable future) qui étendent l'instanciation courante des variables $\langle P_1, \ldots, P_i \rangle$ en une instanciation cohérente (i.e. satisfaisant la contrainte globale PREFIX-PROJECTION $(P, SDB, minsup)$).

PROPOSITION 24 (Valeurs consistantes). — *Soit*[2] *$\sigma = \langle d_1, ..., d_i \rangle$ une instanciation des variables $\langle P_1, \ldots, P_i \rangle$ et P_{i+1} une variable future. Une valeur $d \in D(P_{i+1})$ participe à une solution de* PREFIX-PROJECTION$(P, SDB, minsup)$ *si et seulement si d est un item fréquent dans $SDB|_\sigma$:*

$$|\{(sid, \gamma) | (sid, \gamma) \in SDB|_\sigma \wedge \langle d_{i+1} \rangle \preceq \gamma\}| \geq minsup$$

PREUVE. — Supposons que la valeur $d \in D(P_{i+1})$ apparaisse dans $SDB|_\sigma$ plus de $minsup$ fois. De la proposition 13, nous avons $sup_{SDB}(concat(\sigma, \langle d \rangle)) = sup_{SDB|_\sigma}(\langle d \rangle)$. Par conséquent, l'instanciation $concat(\sigma, \langle d \rangle)$ satisfait la contrainte, et la valeur $d \in D(P_{i+1})$ est donc viable (i.e. participe à au moins une solution). ■

4.3. Prise en compte de l'anti-monotonie de la fréquence

La contrainte globale PREFIX-PROJECTION, outre le calcul de la fréquence des motifs, permet de tirer parti de la propriété d'anti-monotonie de cette contrainte.

2. Nous noterons indifféremment σ soit par $\langle d_1, \ldots, d_i \rangle$ ou par $\langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$.

DÉFINITION 25 (Contrainte anti-monotone). — *Une contrainte c est anti-monotone, si et seulement si, pour tout motif satisfaisant c, tous ses sous-motifs satisfont également c.*

Si un motif séquentiel p satisfait une contrainte anti-monotone, alors toute sous-séquence de p la satisfera aussi. De même, si un motif séquentiel p ne vérifie pas une contrainte anti-monotone, alors aucune sur-séquence de p ne pourra la vérifier. La contrainte de support minimal est une contrainte anti-monotone.

EXEMPLE 26. — Reprenons l'exemple 6 avec $minsup$=2. Le motif $\langle AC \rangle$ satisfait la contrainte de support minimal, et tout sous-motif de $\langle AC \rangle$ vérifiera aussi cette contrainte. Le motif $\langle BA \rangle$ ne satisfait pas la contrainte de support minimal, et aucun sur-motif de $\langle BA \rangle$ ne pourra vérifier cette contrainte. □

La proposition ci-dessous montre comment la contrainte PREFIX-PROJECTION tire parti de l'anti-monotonie de la contrainte de support minimal pour réduire les domaines des variables.

PROPOSITION 27 (Règles de filtrage). — *Soit σ=$\langle d_1, \ldots, d_i \rangle$ l'instanciation courante des variables $\langle P_1, \ldots, P_i \rangle$. Toute valeur $d \in D(P_{i+1})$ qui est localement non fréquente dans $SDB|_\sigma$ peut être retirée du domaine de la variable P_{i+1}. De plus, la valeur d peut être aussi retirée des domaines des variables P_j avec $j \in [i+2, \ldots, \ell]$.*

PREUVE. — Soit $\sigma = \langle d_1, \ldots, d_i \rangle$ l'instanciation courante des variables $\langle P_1, \ldots, P_i \rangle$. Soient $d \in D(P_{i+1})$ et $\sigma' = concat(\sigma, \langle d \rangle)$. Supposons que la valeur d ne soit pas fréquente dans $SDB|_\sigma$. Selon la proposition 13, nous avons $sup_{SDB|_\sigma}(\langle d \rangle) = sup_{SDB}(\sigma') < minsup$. Par conséquent, σ' n'est pas fréquent dans SDB. Donc, la valeur d peut être retirée du domaine de P_{i+1}.

Supposons maintenant que l'instanciation σ soit étendue à $concat(\sigma, \alpha)$, où α correspond à une instanciation quelconque des variables P_j (avec $j > i$). Si la valeur $d \in D(P_{i+1})$ est non fréquente, il est immédiat que $sup_{SDB|_\sigma}(concat(\alpha, \langle d \rangle)) \leq sup_{SDB|_\sigma}(\langle d \rangle) < minsup$. Ainsi, si la valeur d n'est pas fréquente dans $SDB|_\sigma$, elle ne peut être fréquente dans $SDB|_{concat(\sigma,\alpha)}$. Donc, d peut aussi être retirée du domaine des variables P_j pour $j \in [i+2, \ldots, \ell]$. ■

EXEMPLE 28. — Considérons la base de séquences SDB_1 du tableau 1 avec un seuil de support minimal $minsup = 2$. Soit $P = \langle P_1, P_2, P_3 \rangle$ avec $D(P_1) = \mathcal{I}$ et $D(P_2) = D(P_3) = \mathcal{I} \cup \{\square\}$. Supposons que $\sigma(P_1) = A$. PREFIX-PROJECTION$(P, SDB, minsup)$ retirera les deux valeurs A et D des domaines des variables P_2 et P_3 car les seuls items localement fréquents dans $SDB_1|_{<A>}$ sont B et C. □

La contrainte globale PREFIX-PROJECTION tire parti de l'anti-monotonie de la contrainte de support minimal de façon simple et élégante. Ce qui n'est pas le cas des autres approches PPC pour EMS (cf. la section 3.2). Ainsi, (Négrevergne, Guns, 2015) introduit des variables entières supplémentaires (une pour chaque séquence) et utilise des propagateurs et des stratégies de recherche spécifiques, rendant cette intégration plus complexe.

Algorithme 1 : *ProjectSDB* (SDB, $ProjSDB$, α)

Données : SDB : base de séquence initiale ; $ProjSDB$: séquences projetées ; α : préfixe

début

1 $SDB|_\alpha \leftarrow \emptyset$;
2 **pour chaque** *couple* $(sid, start) \in ProjSDB$ **faire**
3 $s \leftarrow SDB[sid]$;
4 $pos_\alpha \leftarrow 1$; $pos_s \leftarrow start$;
5 **tant que** $(pos_\alpha \leq |\alpha| \wedge pos_s \leq |s|)$ **faire**
6 **si** $(\alpha[pos_\alpha] = s[pos_s]$ **alors**
7 $pos_\alpha \leftarrow pos_\alpha + 1$;
8 $pos_s \leftarrow pos_s + 1$;
9 **si** $pos_s \leq |s|$ **alors**
10 $SDB|_\alpha \leftarrow SDB|_\alpha \cup \{(sid, pos_s)\}$
11 **retourner** $SDB|_\alpha$;

4.4. Construction des bases de séquences projetées

Notre approche utilise le principe de pseudo-projection de `PrefixSpan` (cf. la section 2.2). Lorsqu'un préfixe est projeté sur une base de séquences, au lieu de stocker l'ensemble des suffixes sous forme de sous-séquences projetées, chaque suffixe est représenté par un couple $(sid, start)$ où sid est l'identifiant de la séquence et $start$ la position de départ du suffixe dans la séquence sid.

EXEMPLE 29. — Considérons la base de séquences du tableau 1. Comme illustré dans l'exemple 12, $SDB|_{\langle A \rangle}$ est composée de 3 suffixes : $\{(1, \langle BCBC \rangle), (2, \langle BC \rangle), (3, \langle B \rangle)\}$. En utilisant le principe de pseudo projection, $SDB|_{\langle A \rangle}$ peut être représentée par les 3 couples suivants : $\{(1, 2), (2, 3), (3, 2)\}$. □

L'algorithme 1 effectue la pseudo-projection. Il prend en entrées l'ensemble des séquences projetées $ProjSDB$ et un préfixe α. Il parcourt tous les couples $(sid, start)$ de $ProjSDB$ (ligne 2), et cherche à trouver la première position correspondante pour chaque item de α dans la séquence s identifiée par sid. Cette opération est effectuée en incrémentant la position pos_α s'il y a correspondance entre les deux items $\alpha[pos_\alpha]$ et $s[pos_s]$ (lignes 6-7).

Dans le pire des cas, *ProjectSDB* doit parcourir tous les items de toutes les séquences. Par conséquent, la complexité en temps est en $O(\ell \times m)$, où $m = |SDB|$ et ℓ est la longueur de la plus longue séquence dans SDB. La complexité spatiale de la pseudo-projection dans le pire des cas est en $O(m)$, puisque nous avons besoin de stocker au niveau de chaque séquence seulement le couple $(sid, start)$. A noter que, pour la projection standard, la complexité est en $O(m \times \ell)$.

Algorithme 2 : *Filtrage-Prefix-Projection* (SDB, σ, i, P, $minsup$)

Données : SDB : la base initiale de séquences ; σ : le préfixe courant $\langle\sigma(P_1), \ldots, \sigma(P_i)\rangle$; $minsup$: le seuil de support minimal ; $\mathcal{PSDB}$: une structure de données interne utilisée par PREFIX-PROJECTION pour le stockage des pseudo-projections

début

1 **si** $(i \geq 2 \wedge \sigma(P_i) = \square)$ **alors**
2 **pour** $j \leftarrow i+1$ **à** ℓ **faire**
3 $P_j \leftarrow \square$;
4 **retourner** Vrai ;

sinon

5 $\mathcal{PSDB}_i \leftarrow ProjectSDB(SDB, \mathcal{PSDB}_{i-1}, \langle\sigma(P_i)\rangle)$;
6 **si** $(|\,\mathcal{PSDB}_i\,| < minsup)$ **alors**
7 **retourner** Faux ;

sinon

8 $\mathcal{FI} \leftarrow getFreqItems(SDB, \mathcal{PSDB}_i, minsup)$;
9 **pour** $j \leftarrow i+1$ **à** ℓ **faire**
10 **pour chaque** $a \in D(P_j)\ s.t. (a \neq \square \wedge a \notin \mathcal{FI})$ **faire**
11 $D(P_j) \leftarrow D(P_j) - \{a\}$;
12 **retourner** Vrai ;

Fonction getFreqItems (SDB, $ProjSDB$, $minsup$) ;

Données : SDB : la base de séquences initiale ; $ProjSDB$: une pseudo-projection $ExistsItem$, $SupCount$: structure de données internes utilisant une table de hachage pour le calcul de support sur les items;

début

13 $SupCount[] \leftarrow [0, ..., 0]$; $F \leftarrow \emptyset$;
14 **pour chaque** *couple* $(sid, start) \in ProjSDB$ **faire**
15 $ExistsItem[] \leftarrow [faux, ..., faux]$; $s \leftarrow SDB[sid]$;
16 **pour** $i \leftarrow start$ **à** $|s|$ **faire**
17 $a \leftarrow s[i]$;
18 **si** $\neg ExistsItem[a]$ **alors**
19 $SupCount[a] \leftarrow SupCount[a] + 1$;
20 $ExistsItem[a] \leftarrow Vrai$;
21 **si** *(*$SupCount[a] \geq minsup$*)* **alors**
22 $F \leftarrow F \cup \{a\}$;
23 **retourner** F ;

4.5. Algorithme de filtrage pour PREFIX-PROJECTION

Assurer la cohérence de domaine pour la contrainte PREFIX-PROJECTION revient à trouver un motif séquentiel de longueur $(\ell - 1)$, puis à vérifier si ce motif reste fréquent lorsqu'il est étendu avec n'importe quel item d de $D(P_\ell)$. Ainsi, trouver une telle

instanciation est aussi difficile que le problème EMS original. (Yang, 2006) a prouvé que le problème de comptage du nombre de motifs séquentiels maximaux [3] dans une base de séquences est #P-complet, prouvant ainsi la NP-difficulté du problème d'extraction de motifs séquentiels maximaux. La difficulté est due au nombre exponentiel de candidats qui doivent être analysés pour trouver les motifs fréquents. Trouver, pour chaque variable $P_i \in P$ et chaque valeur $d_j \in D(P_i)$, une instanciation σ qui satisfait la contrainte globale t.q. $\sigma(P_i) = d_j$ est donc de complexité exponentielle.

L'algorithme de filtrage de la contrainte PREFIX-PROJECTION maintient une cohérence moins forte que la cohérence de domaine. La cohérence retenue s'appuie sur des propriétés spécifiques des bases projetées (cf. la proposition 24) et sur la propriété d'anti-monotonie du support minimal (cf. la proposition 27). Même si la cohérence maintenue s'apparente à du *forward-checking*, PREFIX-PROJECTION est considérée comme une contrainte globale, car toutes les variables partagent les mêmes structures de données internes qui permettent d'activer et d'effectuer le filtrage.

L'algorithme 2 décrit le pseudo-code de l'algorithme de filtrage de la contrainte PREFIX-PROJECTION. Cet algorithme incrémental est exécuté lorsque les i premières variables sont instanciées. Il est lié à la stratégie de recherche adoptée. De plus, il est particulièrement efficace si les variables sont instanciées selon l'ordre lexicographique : P_1 en premier, ensuite P_2, et ainsi de suite. Il utilise deux structures de données principales permettant d'améliorer ses performances :

– $\mathcal{PSDB}$ stocke les pseudo-projections intermédiaires de SDB, où $\mathcal{PSDB}_i$ ($i \in [0, \ldots, \ell]$) correspond à la σ-projection de l'instanciation partielle courante $\sigma = \langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$ (appelée aussi préfixe) des variables $\langle P_1, \ldots, P_i \rangle$, et $\mathcal{PSDB}_0 = \{(sid, 1) | (sid, s) \in SDB\}$ est la pseudo-projection de la base initiale SDB (i.e. cas où $\sigma = \langle\rangle$).

– Une table de hachage permet d'indexer les items $\mathcal{I}$ vers les entiers $1..|\mathcal{I}|$ afin d'assurer un calcul de support plus efficace sur les items (cf. la fonction `getFreqItems`).

L'algorithme 2 prend en entrée l'instanciation partielle courante σ des variables $\langle P_1, \ldots, P_i \rangle$, la position i de la dernière variable instanciée dans P et le seuil de support minimal $minsup$. Tout d'abord, l'algorithme vérifie si la dernière variable instanciée P_i est différente de $\square$ (cf. la ligne 1). Si c'est le cas, la fin de la séquence est atteinte (car la valeur $\square$ ne peut apparaître qu'à la fin d'une séquence) et la séquence $\langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$ constitue un motif séquentiel dans SDB ; par conséquent, l'algorithme force les $(\ell - i)$ variables restantes non instanciées à la valeur $\square$, puis retourne la valeur *Vrai* (cf. lignes 2-4). Sinon, l'algorithme calcule de manière incrémentale $\mathcal{PSDB}_i$ à partir de $\mathcal{PSDB}_{i-1}$ en invoquant la fonction *ProjectSDB* (cf. l'algorithme 1). Ensuite, il vérifie à la ligne 6 si l'instanciation courante σ est un préfixe fréquent (cf. la proposition 24). Ceci est fait en calculant la taille de $\mathcal{PSDB}_i$. Si cette taille est inférieure à $minsup$, alors l'algorithme arrête d'étendre le préfixe

3. Un motif séquentiel p est maximal s'il n'existe aucun motif q telle que $p \preceq q$.

courant σ et retourne la valeur *Faux*. Dans le cas contraire, l'algorithme calcule l'ensemble des items localement fréquents $\mathcal{FI}$ dans $\mathcal{PSDB}_i$ en invoquant la fonction `getFreqItems` (cf. la ligne 8).

La fonction `getFreqItems` commence par parcourir toutes les entrées de la pseudo-projection une par une, calcule le nombre de premières occurrences d'un item a (i.e. $SupCount[a]$) dans chaque entrée $(sid, start)$, puis garde tous les items fréquents (lignes 13-22). Ceci est fait en utilisant la structure de données $ExistsItem$. Une fois que toutes les entrées de la pseudo-projection ont été considérées, les items fréquents sont retournés (cf. la ligne 23), et l'algorithme 2 met à jour les domaines courants des variables P_j avec $j \geq (i+1)$ en supprimant les valeurs inconsistantes, évitant ainsi d'explorer des branches dans l'arbre de recherche qui ne peuvent mener à des motifs fréquents (cf. les lignes 9-11).

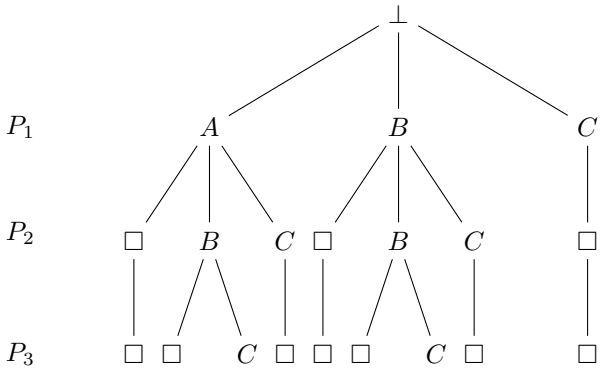


Figure 1. L'espace de recherche associé à l'exemple 30

EXEMPLE 30. — Considérons la base de séquences SDB_1 du tableau 1 avec $minsup = 2$. Soit $P = \langle P_1, P_2, P_3 \rangle$ avec les domaines suivants : $D(P_1) = \mathcal{I}$, $D(P_2) = D(P_3) = \mathcal{I} \cup \{\square\}$. La figure 1 illustre l'espace de recherche exploré lors du filtrage effectué par la contrainte globale PREFIX-PROJECTION. Nous adoptons une stratégie de recherche basée sur l'ordre lexicographique des variables : P_1 est instanciée en premier, ensuite P_2, enfin P_3. Pour la stratégie de choix de valeurs, la plus petite valeur dans le domaine (par rapport à son ordre lexicographique) est choisie en premier. Initialement, avant de lancer l'algorithme *Filter-Prefix-Projection,* les items non fréquents sont retirés des domaines de toutes les variables. Par conséquent, la première variable P_1 sera instanciée à A, B et ensuite C. Puisque le filtrage est basé sur le préfixe courant, il est clair que l'algorithme de filtrage est efficace seulement si au moins la première variable (P_1) est instanciée. Supposons que $\sigma(P_1) = A$, l'algorithme calcule d'abord la base projetée $SDB_1|_{\langle A \rangle}$ à partir de la base initiale SDB_1 (ligne 5) : $\mathcal{PSDB}_1 = \{(1,1),(2,2),(3,1)\}$. Ensuite, il s'assure que l'instanciation courante σ

est consistante (lignes 6-7). Si le préfixe courant σ apparaît dans au moins $minsup$ séquences de SDB_1, les items fréquents permettant d'étendre $\sigma = \langle A \rangle$ vers un motif séquentiel seront calculés (ligne 8). Par conséquent, les deux items non fréquents A et D seront retirés de $D(P_2)$ et $D(P_3)$ (cf. l'exemple 28). Il y a un cas particulier à traiter lorsqu'une variable est instanciée au symbole $\square$. Par exemple, on suppose qu'après instanciation de P_1 à A, la variable P_2 est instanciée à $\square$. Puisque ce symbole désigne la fin de la séquence, toutes les variables qui suivent P_2 doivent être instanciées aussi à $\square$ (lignes 1-3). Nous obtenons ainsi le motif séquentiel $\langle A\square\square \rangle = \langle A \rangle$. $\square$

La proposition 31 affirme que l'algorithme *Filter-Prefix-Projection* établit la cohérence de domaine sur la variable P_{i+1} qui suit le préfixe courant.

PROPOSITION 31 (Cohérence de domaine de la prochaine variable). — *Soient $\sigma = \langle d_1, \ldots, d_i \rangle$ une instanciation partielle consistante de i variables $\langle P_1, \ldots, P_i \rangle$ (i.e., le préfixe courant fréquent), et $P_{i+1} \in P$ une variable libre (i.e. non instanciée). L'algorithme Filter-Prefix-Projection assure la cohérence de domaine sur la variable P_{i+1} pour la contrainte* PREFIX-PROJECTION.

PREUVE. — Puisque σ est consistant, alors à partir de la proposition 23, nous avons $|SDB|_\sigma| \geq minsup$. L'algorithme *Filter-Prefix-Projection* retire à partir du domaine de P_{i+1} toutes les valeurs qui ne sont pas fréquentes dans $SDB|_\sigma$. Il garde ainsi que les valeurs consistantes fournies par la proposition 24. Par conséquent, *Filter-Prefix-Projection* assure la cohérence de domaine sur la variable P_{i+1}. ■

PROPOSITION 32 (Complexité du filtrage). — *Dans le pire des cas, le filtrage de la contrainte* PREFIX-PROJECTION *peut être réalisé en $O(m \times \ell + m \times d + \ell \times d)$. La complexité en espace de* PREFIX-PROJECTION *est en $O(m \times \ell)$.*

PREUVE. — Soit ℓ la longueur de la plus longue séquence de SDB, $m = |SDB|$, et $d = |\mathcal{I}|$. Le calcul de la pseudo-projection de $\mathcal{PSDB}_i$ s'effectue en $O(m \times \ell)$: pour chaque séquence (sid, s) de SDB, vérifier si σ apparaît dans s se fait en $O(\ell)$ et il y a m séquences. La complexité totale de la fonction *GetFreqItems* est $O(m \times (\ell + d))$. Les lignes (9-11) s'effectuent en $O(\ell \times d)$. Ainsi, la complexité finale est en $O(m \times \ell + m \times (\ell + d) + \ell \times d)$ i.e. $O(m \times \ell + m \times d + \ell \times d)$. La complexité en espace de l'algorithme de filtrage est liée à la structure de données interne $\mathcal{PSDB}$. Dans le pire des cas, ℓ bases projetées doivent être stockées. Comme chaque pseudo-projection d'une base nécessite $O(m)$, la complexité finale dans le pire des cas est $O(m \times \ell)$. ■

La proposition suivante montre que la recherche de solutions peut se faire en temps polynomial en fonction du nombre de solutions N.

PROPOSITION 33 (Complexité de la recherche des solutions). — *Supposons que les variables $\langle P_1, \ldots, P_\ell \rangle$ sont énumérées suivant l'ordre lexicographique $P_1 \rightarrow P_2 \rightarrow \cdots \rightarrow P_\ell$. L'extraction de l'ensemble de tous les motifs séquentiels de cardinalité N, se fait sans échec, et en $O(N \times \ell \times (m \times \ell + m \times d + \ell \times d))$.*

PREUVE. — En utilisant la proposition 31, l'ordre adopté des variables garantit la cohérence de domaine de la prochaine variable. En conséquence, la recherche des

solutions se fait sans échec. Etant donné qu'il n'y a pas d'échec, le nombre de feuilles de l'arbre de recherche est N, et la taille de l'arbre de recherche est borné par $O(N \times \ell)$. Au niveau de chaque nœud de l'arbre de recherche, le solveur fait appel uniquement à l'algorithme de filtrage *Filter-Prefix-Projection* . Ainsi, nous avons la complexité donnée en fonction du nombre de solutions N. ■

En conséquence de la proposition 31, pour avoir une bonne performance de la recherche des solutions, il est nécessaire d'avoir autant de nœuds possibles où toutes les variables qui précèdent la variable à instancier, sont déjà instanciées. Le comportement idéal est celui de l'ordre lexicographique où la cohérence de domaine est assurée sur la prochaine variable.

4.6. Comparaison entre PREFIX-PROJECTION *et* GLOBAL-P.F

Dans cette section, nous comparons notre approche basée sur la contrainte globale PREFIX-PROJECTION et le modèle reifié de (Négrevergne, Guns, 2015) (détaillé en section 3.2) au niveau modélisation et capacité de filtrage. Par exemple, considérons la base de séquences du tableau 1 avec $minsup = 3$. Soit $P = \langle P_1, P_2, P_3, P_4, P_5 \rangle$. On suppose maintenant que $\sigma(P_1) = B$ et $D(P_2) = D(P_3) = D(P_4) = D(P_5) = \mathcal{I} \cup \{\square\}$.

Le tableau 2 compare le modèle PPC des deux approches. Comme indiqué dans le tableau, notre approche nécessite seulement une seule contrainte et quatre variables afin de modéliser la relation de sous-séquence combinée avec la contrainte de fréquence. En revanche, le modèle réifié utilise 8 contraintes (i.e., $C_1, ..., C_8$) et 8 variables additionnelles : 4 variables booléennes S_1, ..., S_4 (une pour chaque contrainte globale `exists-embedding`), et 4 variables auxiliaires X_1, ..., X_4 où chaque X_i ($i \in \{1, ..., 4\}$) stocke les items qui apparaissent après le préfixe courant dans la séquence S_i. Dans ce qui suit, nous allons détailler le processus de filtrage et la réduction des domaines établis par les deux modèles sur les domaines courants de l'ensemble des variables :

GLOBAL-P.F :

– Pour les contraintes de *fin de séquences* (voir ligne 1 dans les contraintes associées à GLOBAL-P.F), comme aucune variable P_j n'est instanciée à $\square$, aucune réduction de domaine n'est faite sur les variables P.

– De même, aucun filtrage n'est effectué par les contraintes d'*inclusion*. Le domaine de S_i ne peut pas être réduit puisque $\langle B \rangle$ apparaît dans toutes les séquences. Par conséquent, aucun filtrage ne peut être établi sur les variables P. Chaque variable auxiliaire X_i ($i \in 1, ..., 4$), prend comme domaine les items qui apparaissent après le préfixe courant $\langle B \rangle$ comme suit : $D(X_1) = \{B, C, \square\}$, $D(X_2) = \{A, B, C, \square\}$, $D(X_3) = \{\square\}$, $D(X_4) = \{C, D, \square\}$. Nous signalons que les valeurs des domaines des variables X sont exploitées par une routine de branchement spécialisée (choix de valeurs) afin d'éviter l'exploration des items non fréquents. Dans cet exemple, il y a

que l'item C qui est fréquent, donc, pour la prochaine variable non instanciée, la routine de branchement va permettre de brancher obligatoirement sur cet item fréquent.

– La contrainte de fréquence ne filtre pas, puisque les variables S ne sont pas instanciées.

– Finalement, GLOBAL-P.F ne fait aucune réduction de domaine sur les variables P.

PREFIX-PROJECTION :

En exécutant l'algorithme de filtrage *Filter-Prefix-Projection,* plusieurs valeurs inconsistantes seront élaguées à partir des domaines des variables P. Nous obtenons ainsi les domaines suivants : $P_i \in \{C, \square\}$ $(i = 2, ..., 5)$.

Tableau 2. Modèles PPC pour EMS : PREFIX-PROJECTION *vs* GLOBAL-P.F

	PREFIX-PROJECTION	GLOBAL-P.F
Variables & Domaines	$P = \langle P_1, P_2, \ldots, P_5 \rangle$ % motif inconnu $D(P_1) = \{B\}$ $D(P_2) = \{A, B, C, D, \square\}$ $D(P_3) = \{A, B, C, D, \square\}$ $D(P_4) = \{A, B, C, D, \square\}$ $D(P_5) = \{A, B, C, D, \square\}$	$P = \langle P_1, P_2, \ldots, P_5 \rangle$ % motif inconnu $D(P_1) = \{B\}$ $D(P_2) = D(P_3) = D(P_4) = D(P_5) = \{A, B, C, D, \square\}$ $S = \langle S_1, S_2, S_3, S_4 \rangle$ % variables booléennes pour assurer les contraintes de couverture $X = \langle X_1, X_2, X_3, X_4 \rangle$ % variables auxiliaires pour éviter les symboles non-fréquents $D(X_1) = D(X_2) = D(X_3) = D(X_4) = \{A, B, C, D, \square\}$
Contraintes	PREFIX-PROJECTION$(P, SDB, minsup)$	$\forall j \in 2 \ldots 4, C_{j-1} : P_j = \square \rightarrow P_{j+1} = \square$ % contraintes de fin de séquence C_4 : exists-embedding$(P, SDB[1], S_1, X_1)$ % contraintes d'inclusion C_5 : exists-embedding$(P, SDB[2], S_2, X_2)$ % en ce sens $S_i \longleftrightarrow \exists e$ s .t.$P \preceq SDB[i]$ C_6 : exists-embedding$(P, SDB[3], S_3, X_3)$ C_7 : exists-embedding$(P, SDB[4], S_4, X_4)$ $C_8 : S_1 + S_2 + S_3 + S_4 \geq 3$ % contrainte de fréquence

4.7. Encodage des contraintes usuelles sur les motifs

Les contraintes définies dans la section 2.3 peuvent être formulées de manière simple en utilisant directement les contraintes globales du solveur. Soit P le motif inconnu recherché de longueur ℓ.

– Contrainte de longueur minimale : $minSize(P, \ell_{min}) \equiv \bigwedge_{i=1}^{i=\ell_{min}} (P_i \neq \square)$

– Contrainte de longueur maximale : $maxSize(P, \ell_{max}) \equiv \bigwedge_{i=\ell_{max}+1}^{i=\ell} (P_i = \square)$

– Contrainte d'appartenance d'items : soient V un sous-ensemble d'items, l et u deux entiers tels que $0 \leq l \leq u \leq \ell$. Alors, $item(P, V) \equiv \bigwedge_{t \in V} \texttt{Among}(P, \{t\}, l, u)$. Cette contrainte impose que les items de V doivent être présents au moins l fois et au plus u fois dans P. Pour interdire l'apparition d'un item dans P, il suffit de mettre l et u à 0.

– Contrainte d'expression régulière : soit A_{reg} l'automate déterministe associé à l'expression régulière exp. Alors, $reg(P, exp) \equiv \texttt{Regular}(P, A_{reg})$.

5. Expérimentations

Nos expérimentations ont été menées sur plusieurs jeux de données réels obtenus à partir de (Fournier-Viger *et al.*, 2014 ; Béchet *et al.*, 2012 ; Trasarti *et al.*, 2008), et

Tableau 3. Caractéristiques des jeux de données utilisés

Jeu de données	$\|SDB\|$	$\|\mathcal{I}\|$	avg $\|s\|$	$\max_{s \in SDB} \|s\|$	type de données
Leviathen	5 834	9 025	33,81	100	livre
Kosarak	69 999	21 144	7,97	796	flux de clics web
FIFA	20 450	2 990	34,74	100	flux de clics web
BIBLE	36 369	13 905	21,64	100	bible
Protein	103 120	24	482	600	séquences de protéines
data-200K	200 000	20	50	86	base de séquences synthétique
PubMed	17 527	19 931	29	198	texte bio-médical

représentant différents domaines d'application. Les différentes caractéristiques de ces jeux de données sont résumées dans le tableau 3.

L'objectif de ces expérimentations est double :

1. Comparer notre approche aux approches PPC existantes ainsi qu'aux méthodes de l'état de l'art pour le problème EMS en termes de passage à l'échelle.

2. Montrer la flexibilité de notre approche qui permet de traiter et d'imposer simultanément plusieurs classes de contraintes.

5.1. Protocole expérimental

Notre approche (notée `PP`) utilise le solveur de contraintes `gecode` [4]. Toutes les expérimentations ont été menées sur une machine avec un processeur Intel X5670 ayant 24 GB de mémoire. Nous avons utilisé une limite de temps d'une heure. Pour chaque base de séquences, nous avons fait varier le seuil de support minimal $minsup$ jusqu'à ce qu'on ne parvienne plus à terminer l'extraction de tous les motifs dans le délai imparti (1 heure). Les paramètres ℓ_{min} et ℓ ont été fixés respectivement à 1 et à la taille de la plus grande séquence dans SDB. Nous avons comparé [5] notre approche `PP` avec :

1. Les deux meilleures approches PPC existantes pour le problème EMS (Négrevergne, Guns, 2015) : GLOBAL-P.F et DECOMPOSED-P.F (cf. la section 3.2).

2. Deux méthodes spécialisées de l'état de l'art pour le problème EMS : `PrefixSpan` et `cSpade` (cf. la section 3.1) ;

3. La méthode `SMA` (Trasarti *et al.*, 2008) pour l'extraction de motifs séquentiels sous contraintes d'expressions régulières (cf. la section 3.1).

4. http://www.gecode.org

5. Code source et jeux de données sont disponibles à l'url https://sites.google.com/site/prefixprojection4cp/

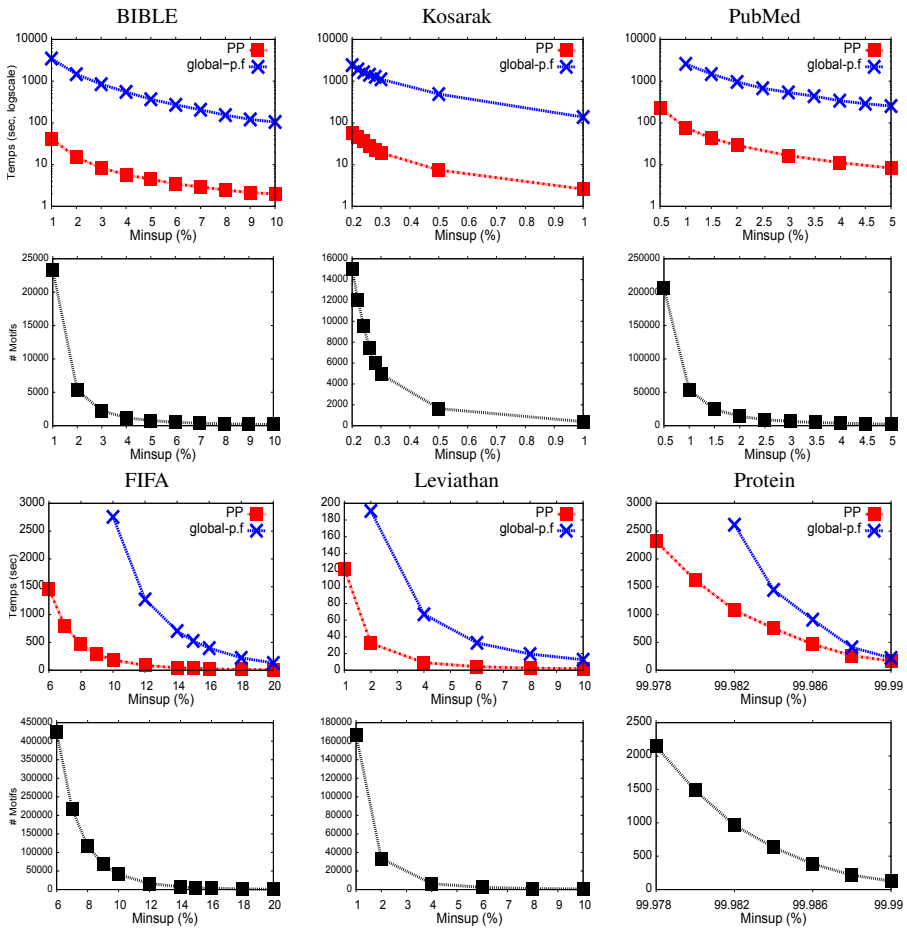


Figure 2. `PP` *vs* GLOBAL-P.F *Temps CPU (haut) et nombre de motifs extraits (bas)*

Nous avons utilisé l'implémentation de `PrefixSpan` réalisée par Y. Tabei [6] et les implémentations de `cSpade` [7] et de `SMA` [8] fournies par les auteurs. L'implémentation [9] des deux approches PPC utilise aussi le solveur de contraintes `gecode`.

5.2. Comparaison avec les approches PPC pour EMS

Cette section compare `PP` avec les deux approches retenues : GLOBAL-P.F et

6. https://code.google.com/p/prefixspan/
7. http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/
8. http://www-kdd.isti.cnr.it/SMA/
9. https://dtai.cs.kuleuven.be/CP4IM/cpsm/

DECOMPOSED-P.F (cf. la section 3.2). La figure 2 indique le nombre de motifs séquentiels extraits et les temps CPU nécessaires pour seulement les deux méthodes PP et GLOBAL-P.F puisque DECOMPOSED-P.F n'arrive pas à terminer l'extraction avant le délai fixé à une heure (une échelle logarithmique est utilisée pour BIBLE, Kosarak et PubMed). Des résultats de la figure 2, nous pouvons dresser les remarques suivantes :

– Comme attendu, le nombre de motifs extraits diminue avec l'augmentation du seuil de support minimal $minsup$.

– En comparant les temps CPU, DECOMPOSED-P.F est la méthode la moins performante (elle n'apparait pas dans la figure) : sur toutes les bases de séquences, elle ne parvient pas à terminer l'extraction de tous les motifs séquentiels dans un délai d'une heure et cela pour toutes les valeurs de $minsup$ considérées.

– PP domine largement GLOBAL-P.F sur toutes les bases : PP est au moins un ordre de grandeur plus rapide que GLOBAL-P.F. Les gains en termes de temps CPU sont grandement amplifiés pour les petites valeurs de $minsup$. Sur BIBLE (resp. PubMed), le *speedup* est de $84,4$ (resp. $33,5$) pour une valeur de $minsup$ égale à $1\ \%$.

– Enfin, sur la plupart des bases de séquences (à l'exception de BIBLE et Kosarak), GLOBAL-P.F n'est pas en mesure d'extraire tous les motifs séquentiels pour de faibles valeurs de $minsup$ dans un délai d'une heure. Par exemple, sur FIFA, PP extrait l'ensemble de tous les motifs séquentiels pour des valeurs de $minsup$ allant jusqu'à $6\ \%$ en $1\ 457$ secondes, tandis que GLOBAL-P.F ne parvient pas à terminer l'extraction pour des valeurs de $minsup$ inférieures à $10\ \%$.

Pour compléter ces premiers résultats décrits dans la figure 2, le tableau 4 donne sur différentes bases de séquences et différentes valeurs de $minsup$, le nombre d'appels à la routine de propagation du solveur Gecode (cf. colonne 5), et le nombre de nœuds de l'arbre de recherche (cf. colonne 6). Premièrement, PP explore moins de nœuds que GLOBAL-P.F. Toutefois, la différence n'est pas très importante (un gain d'environ 45 % et 33 % sur FIFA et BIBLE respectivement). Deuxièmement, notre approche est très efficace en termes de nombre de propagations effectuées. Pour PP, le nombre de propagations reste faible (en milliers pour des valeurs petites de $minsup$) par rapport à GLOBAL-P.F (en millions). Cela est dû au très grand nombre de contraintes réifiées utilisées par GLOBAL-P.F pour encoder la relation de sous-séquence. A l'opposé, la contrainte globale PREFIX-PROJECTION ne requiert aucune contrainte réifiée ni aucune variable supplémentaire pour encoder la relation de sous-séquence.

5.3. Comparaison avec les méthodes spécialisées pour EMS

Cette section compare PP avec deux méthodes spécialisées : PrefixSpan et cSpade. En effet, cSpade fait partie des méthodes les plus performantes pour l'extraction de motifs séquentiels sous contraintes. De plus, la comparaison avec PrefixSpan va permettre d'évaluer l'apport de la PPC puisque les deux approches ex-

Tableau 4. PP *vs* GLOBAL-P.F

Base	$minsup$ (%)	#MOTIFS	Temps CPU (s)		#PROPAGATIONS		#NŒUDS	
			PP	GLOBAL-P.F	PP	GLOBAL-P.F	PP	GLOBAL-P.F
FIFA	20	938	**8,16**	129,54	**1 884**	11 649 290	**1 025**	1 873
	18	1 743	**13,39**	222,68	**3 502**	19 736 442	**1 922**	3 486
	16	3 578	**24,39**	396,11	**7 181**	35 942 314	**3 923**	7 151
	14	7 313	**44,08**	704	**14 691**	65 522 076	**8 042**	14 616
	12	16 323	**86,46**	1 271,84	**32 820**	126 187 396	**18 108**	32 604
	10	40 642	**185,88**	2 761,47	**81 767**	266 635 050	**45 452**	81 181
BIBLE	10	174	**1,98**	105,01	**363**	4 189 140	**235**	348
	8	274	**2,47**	153.61	**575**	5 637 671	**362**	548
	6	508	**3,45**	270.49	**1 065**	8 592 858	**669**	1 016
	4	1 185	**5,7**	552,62	**2 482**	15 379 396	**1 575**	2 371
	2	5 311	**15,05**	1 470,45	**11 104**	39 797 508	**7 048**	10 605
	1	23 340	**41,4**	3 494,27	**49 057**	98 676 120	**31 283**	46 557
PubMed	5	2 312	**8,26**	253,16	**4 736**	15 521 327	**2 833**	4 619
	4	3 625	**11,17**	340,24	**7 413**	20 643 992	**4 428**	7 242
	3	6 336	**16,51**	536,96	**12 988**	29 940 327	**7 757**	12 643
	2	13 998	**28,91**	955,54	**28 680**	50 353 208	**17 145**	27 910
	1	53 818	**77,01**	2 581,51	**110 133**	124 197 857	**65 587**	107 051
Protein	99,99	127	**165,31**	219,69	**264**	26 731 250	**172**	221
	99,988	216	**262,12**	411,83	**451**	44 575 117	**293**	390
	99,986	384	**467,96**	909,47	**805**	80 859 312	**514**	679
	99,984	631	**753,3**	1 443,92	**1 322**	132 238 827	**845**	1 119
	99,982	964	**1 078,73**	2 615	**2 014**	201 616 651	**1 284**	1 749
	99,98	2 143	**2 315,65**	–	**4 485**	–	**2 890**	–
Kosarak	1	384	**2.59**	137,95	**793**	8 741 452	**482**	769
	0,5	1 638	**7,42**	491,11	**3 350**	26 604 840	**2 087**	3 271
	0,3	4 943	**19,25**	1 111,16	**10 103**	56 854 431	**6 407**	9 836
	0,28	6 015	**22,83**	1 266,39	**12 308**	64 003 092	**7 831**	11 954
	0,24	9 534	**36,54**	1 635,38	**19 552**	81 485 031	**12 667**	18 966
	0,2	15 010	**57,6**	2 428,23	**30 893**	111 655 799	**20 055**	29 713
Leviathan	10	651	**1,78**	12,56	**1 366**	2 142 870	**849**	1 301
	8	1 133	**2,57**	19,44	**2379**	3 169 615	**1 487**	2 261
	6	2 300	**4,27**	32,85	**4824**	5 212 113	**3 008**	4 575
	4	6 286	**9,08**	66,31	**13197**	10 569 654	**8 227**	12 500
	2	33 387	**32,27**	190,45	**70 016**	33 832 141	**43 588**	66 116
	1	167 189	**121,89**	–	**350 310**	–	**217 904**	–

ploitent le même principe de la projection préfixée. La figure 3 compare les temps d'exécution des trois méthodes.

Premièrement, `cSpade` surclasse les deux autres méthodes sur toutes les bases de séquences (à l'exception de Protein). Malgré tout, `PP` a un comportement relativement similaire à celui de `cSpade`, mais il est moins rapide (sans compter les valeurs élevées de $minsup$). Le comportement de `cSpade` sur Protein est dû à la représentation verticale des bases de séquences qui n'est pas appropriée dans le cas de bases contenant de longues séquences et un nombre d'items relativement faible, dégradant ainsi les performances du processus d'extraction.

Deuxièmement, `PP` qui utilise également le principe de la projection préfixée, surpasse clairement `PrefixSpan` sur toutes les bases. Cela peut s'expliquer par notre algorithme de filtrage qui, combiné avec des structures de données incrémentales, permet une gestion plus efficace des bases de séquences projetées. Sur FIFA, `PrefixSpan` n'est pas en mesure de terminer l'extraction de tous les motifs séquentiels pour $minsup \leq 12\ \%$, alors que notre approche reste efficace jusqu'à $6\ \%$ dans

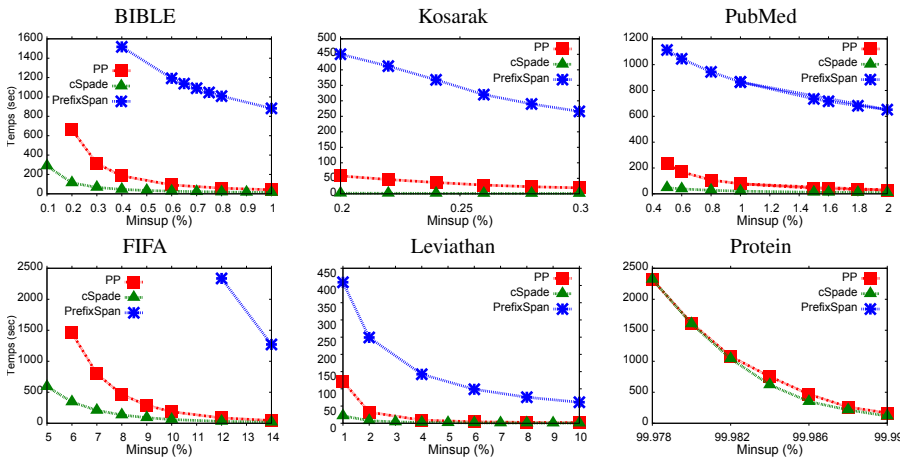


Figure 3. Comparaison de `PP` *avec* `PrefixSpan` *et* `cSpade` *pour EMS*

le délai d'une heure. Sur Protein, `PrefixSpan` n'arrive pas à terminer l'extraction de tous les motifs séquetiels et cela pour toutes les valeurs de $minsup$ que nous avons considérées.

Ces résultats montrent clairement que notre approche est très compétitive vis à vis des meilleures méthodes spécialisées pour l'extraction de motifs séquentiels, et ceci sur des grandes bases de séquences.

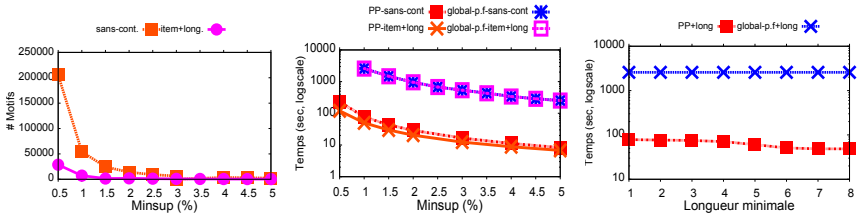
5.4. EMS sous contraintes d'item et de longueur minimale

Cette section illustre l'intérêt de pouvoir utiliser simultanément différents types de contraintes. Nous avons utilisé la base PubMed, et pour extraire des connaissances linguistiques, nous avons combiné différents types de contraintes, telles que la fréquence minimale, la longueur minimale et l'appartenance d'items.

La base de séquences PubMed est construite à partir d'un corpus d'apprentissage. Les séquences sont les phrases du corpus contenant au moins une maladie rare (MR) et un gène. Les maladies rares et les gènes sont remplacés respectivement par les deux items uniques DISEASE et GENE. Le but est d'obtenir des motifs séquentiels qui traduisent certaines régularités linguistiques (c.-à-d. des relations gène-MR) (Béchet *et al.*, 2012).

– La contrainte de longueur minimale permet d'ignorer les motifs jugés trop petits en termes de nombre d'items (nombre de mots) puisque de tels motifs ne peuvent être pertinents. Pour cela, le seuil ℓ_{min} a été fixé à 3.

– La contrainte d'appartenance impose que tout motif extrait contienne au moins les items $GENE$ et $DISEASE$.



(a) # motifs (b) Temps CPU (c) Contrainte de longueur minimale

Figure 4. `PP` *vs* GLOBAL-P.F *sous contraintes de longueur minimale et d'item*

Tableau 5. `PP` *vs* GLOBAL-P.F *sous contraintes de longueur minimale et d'item*

Base	$minsup$ (%)	#MOTIFS	Temps CPU (s)		#PROPAGATIONS		#NOEUDS	
			`PP`	GLOBAL-P.F	`PP`	GLOBAL-P.F	`PP`	GLOBAL-P.F
PubMed	5	279	**6,76**	252,36	**7 878**	12 234 292	**2 285**	4 619
	4	445	**8,81**	339,09	**12 091**	16 475 953	**3 618**	7 242
	3	799	**12,35**	535,32	**20 268**	24 380 096	**6 271**	12 643
	2	1 837	**20,41**	953,32	**43 088**	42 055 022	**13 888**	27 910
	1	7 187	**49,98**	2 574,42	**157 899**	107 978 568	**52 508**	107 051

Comme il n'existe aucune méthode spécialisée qui permette cette combinaison de contraintes, nous avons uniquement comparé `PP` avec GLOBAL-P.F. La figure 4 indique les temps CPU et le nombre de motifs séquentiels extraits avec et sans ces contraintes, ceci pour différentes valeurs de $minsup$.

Premièrement, imposer simultanément les deux contraintes (i.e. contraintes de longueur minimale et d'item) permet de réduire de façon significative le nombre de motifs ainsi que les temps CPU pour `PP`. Les performances de GLOBAL-P.F n'ont pas beaucoup changé en intégrant les contraintes de longueur et de présence d'item. Cela est nécessairement dû au fait que l'approche GLOBAL-P.F est déjà handicapée par le filtrage faible induit par les m contraintes `exists-embedding`.

Deuxièmement (cf. le tableau 5), lorsqu'on considère les deux contraintes *longueur minimale* et *item*, `PP` domine clairement GLOBAL-P.F (*speedup* allant jusqu'à $51,5$). De plus, le nombre de propagations effectuées par `PP` reste petit par rapport à `globalp.f`.

La figure 4c compare les deux méthodes sous la contrainte de longueur minimale pour différentes valeurs de ℓ_{min}. Nous avons fixé $minsup$ à 1 %. `PP` demeure la méthode la plus performante (*speedup* de $53,1$). Ces résultats confirment ce que nous avons observé précédemment, à savoir le filtrage faible induit par les m contraintes globales `exists-embedding` et les contraintes portant sur les motifs.

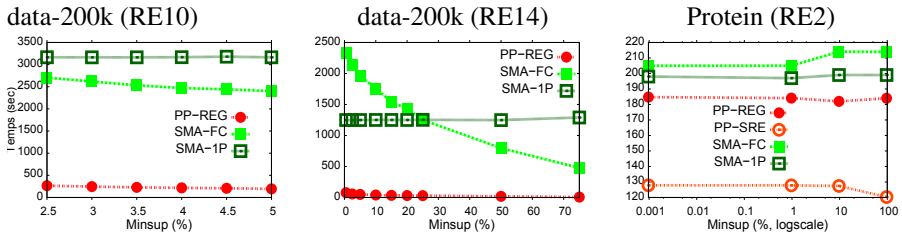


Figure 5. `PP` *vs* `SMA` *sous contraintes d'expression régulière*

5.5. EMS sous contraintes d'expression régulière

Notre dernière série d'expérimentations compare `PP-REG` avec deux variantes de `SMA` : `SMA-1P` et `SMA-FC`. Deux jeux de données sont considérés (Trasarti *et al.*, 2008) : une base synthétique de séquences (data-200k) et une base réelle de séquences (Protein) (cf. le tableau 3).

– Pour la base data-200k, nous avons utilisé les deux expressions régulières suivantes : RE10 $\equiv A^*B(B|C)D^*EF^*(G|H)I^*$ et RE14 $\equiv A^*(Q|BS^*(B|C))D^*E(I|S)^*(F|H)\,G^*R$.

– Pour la base Protein, nous avons utilisé l'expression RE2 $\equiv (S|T).(R|K)$ représentant la Protéine dite *kinase C catalysant la phosphorylation* (où . désigne n'importe quel symbole).

La figure 5 compare les temps de calcul des différentes approches. Sur la base synthétique, notre approche est très efficace. En considérant l'expression régulière RE14, notre approche est un ordre de grandeur plus rapide que `SMA`. Sur la base Protein, l'écart entre les 3 méthodes est réduit, mais notre méthode reste la plus efficace. Pour le cas particulier de l'expression régulière RE2, la contrainte `Regular` peut être remplacée en imposant deux contraintes pour restreindre le domaine de la première (resp. troisième) variable à $\{S, T\}$ (resp. $\{R, K\}$). Cette nouvelle modélisation, désignée par `PP-SRE` sur la figure 5 (partie droite), permet encore d'augmenter l'efficacité.

6. Conclusion

Nous avons proposé dans cet article la contrainte globale PREFIX-PROJECTION pour l'extraction des motifs séquentiels. PREFIX-PROJECTION utilise un encodage concis et son filtrage exploite le principe de *la projection préfixée* de `PrefixSpan`. Puis, nous avons montré comment notre encodage permet de mettre en œuvre différents types de contraintes (appartenance d'items, taille et expressions régulières) et de les combiner simultanément. Par rapport aux approches PPC existantes, notre contrainte globale ne requiert aucune contrainte réifiée ni aucune variable intermédiaire pour encoder la relation de sous-séquence. Enfin, les expérimentations menées montrent que notre approche surpasse les approches PPC existantes et concurrence les méthodes spécialisées sur de grandes bases de séquences.

Les expérimentations montrent que notre approche permet de passer à l'échelle, alors qu'il s'agit d'un enjeu majeur pour les approches PPC existantes.

Ce travail ouvre de nombreuses nouvelles perspectives pour l'extraction de motifs plus riches tels que les motifs fermés, les sous-groupes pertinents ou les skypatterns. Nous comptons également prendre en compte les contraintes sur la forme du motif, notamment la contrainte gap.

Bibliographie

Agrawal R., Srikant R. (1995). Mining sequential patterns. In P. S. Yu, A. L. P. Chen (Eds.), *Proceedings of the eleventh international conference on data engineering (icde)*, p. 3-14. IEEE Computer Society.

Ayres J., Flannick J., Gehrke J., Yiu T. (2002). Sequential pattern mining using a bitmap representation. In *KDD 2002*, p. 429–435. ACM.

Béchet N., Cellier P., Charnois T., Crémilleux B. (2012). Sequential pattern mining to discover relations between genes and rare diseases. In *Cbms.*

Beldiceanu N., Contejean E. (1994). Introducing global constraints in CHIP. *Journal of Mathematical and Computer Modelling*, vol. 20, n° 12, p. 97-123.

Coquery E., Jabbour S., Saïs L., Salhi Y. (2012). A SAT-based approach for discovering frequent, closed and maximal patterns in a sequence. In *Ecai 2012 - 20th european conference on artificial intelligence*, vol. 242, p. 258-263. IOS Press.

Dong G., Pei J. (2007). *Sequence data mining* (vol. 33). Kluwer.

Fournier-Viger P., Gomariz A., Gueniche T., Soltani A., Wu C., Tseng V. (2014). SPMF: A Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research*, vol. 15, p. 3389-3393. Consulté sur http://jmlr.org/papers/v15/fournierviger14a.html

Garofalakis M. N., Rastogi R., Shim K. (2002). Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowl. Data Eng.*, vol. 14, n° 3, p. 530-552.

Guns T., Nijssen S., Raedt L. D. (2011). Itemset mining: A constraint programming perspective. *Artif. Intell.*, vol. 175, n° 12-13, p. 1951-1983.

Kemmar A., Loudni S., Lebbah Y., Boizumault P., Charnois T. (2015). PREFIX-PROJECTION global constraint for sequential pattern mining. In *Principles and practice of constraint programming - 21st international conference, CP 2015, cork, ireland, august 31 - september 4, 2015, proceedings, lncs 9255*, p. 226–243. Consulté sur http://dx.doi.org/10.1007/978-3-319-23219-5_17

Kemmar A., Loudni S., Lebbah Y., Boizumault P., Charnois T. (2016). A global constraint for mining sequential patterns with GAP constraint. In *Integration of AI and OR techniques in constraint programming - 13th international conference, CPAIOR 2016, banff, ab, canada, may 29 - june 1, 2016, proceedings*, vol. 9676, p. 198–215. Springer.

Kemmar A., Ugarte W., Loudni S., Charnois T., Lebbah Y., Boizumault P. *et al.* (2014). Mining relevant sequence patterns with cp-based framework. In *26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014)*, p. 552–559. IEEE Computer Society.

Li C., Yang Q., Wang J., Li M. (2012, mars). Efficient mining of gap-constrained subsequences and its various applications. *ACM Trans. Knowl. Discov. Data*, vol. 6, n° 1, p. 2:1–2:39.

Métivier J.-P., Loudni S., Charnois T. (2013). A constraint programming approach for mining sequential patterns in a sequence database. In *Ecml/pkdd workshop on languages for data mining and machine learning.*

Négrevergne B., Guns T. (2015). Constraint-based sequence mining using constraint programming. In *Integration of AI and OR techniques in constraint programming - 12th international conference, CPAIOR 2015, barcelona, spain, may 18-22, 2015, proceedings*, vol. 9075, p. 288–305. Springer.

Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U. *et al*. (2001). PrefixSpan: Mining sequential patterns by prefix-projected growth. In *Icde*, p. 215-224. IEEE Computer Society.

Pei J., Han J., Wang W. (2002). Mining sequential patterns with constraints in large databases. In *Cikm'02*, p. 18–25. ACM.

Pesant G. (2004). A regular language membership constraint for finite sequences of variables. In M. Wallace (Ed.), *Principles and practice of constraint programming - cp 2004*, vol. 3258, p. 482-495. Springer.

Srikant R., Agrawal R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Edbt*, p. 3-17.

Trasarti R., Bonchi F., Goethals B. (2008). Sequence mining automata: A new technique for mining frequent sequences under regular expressions. In *Icdm'08*, p. 1061-1066.

Wang J., Han J. (2004). BIDE: Efficient mining of frequent closed sequences. In *Icde*, p. 79-90.

Yan X., Han J., Afshar R. (2003). CloSpan: Mining closed sequential patterns in large databases. In D. Barbará, C. Kamath (Eds.), *Proceedings of the third siam international conference on data mining, san francisco (sdm).* SIAM.

Yang G. (2006). Computational aspects of mining maximal frequent patterns. *Theor. Comput. Sci.*, vol. 362, n° 1-3, p. 63–85.

Zaki M. J. (2000). Sequence mining in categorical domains: Incorporating constraints. In *Proceedings of the 2000 ACM CIKM international conference on information and knowledge management, mclean, va, usa, november 6-11, 2000*, p. 422–429.

Zaki M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, vol. 42, n° 1/2, p. 31-60.