HAL
archives-ouvertes.fr

Calcul par Contraintes de Motifs Ordonnés

Vincent Vigneron, David Lesaint, Barry Hurley, Deepak Mehta, Barry O'Sullivan

▶ **To cite this version:**

Vincent Vigneron, David Lesaint, Barry Hurley, Deepak Mehta, Barry O'Sullivan. Calcul par Contraintes de Motifs Ordonnés. 12èmes Journées Francophones de la Programmation par Contraintes (JFPC), 2016, Montpellier, France. pp.143-152. hal-02709490

HAL Id: hal-02709490
https://hal.univ-angers.fr/hal-02709490

Submitted on 1 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Calcul par Contraintes de Motifs Ordonnés

Vincent Vigneron[1] *__David Lesaint__[1] **Barry Hurley**[2] **Deepak Mehta**[2] **Barry O'Sullivan**[2]

[1] LERIA, Université d'Angers, France
[2] INSIGHT Centre for Data Analytics, University College Cork, Ireland
{firstname.lastname}@{univ-angers.fr, insight-centre.org}

Résumé

La recherche de patrons ou d'itemsets fréquents a de nombreuses applications allant de la bioinformatique au marketing. Nous présentons le langage MMP (Maximal Matrix Problem) pour modéliser ces problèmes par une matrice de variables à domaines finis et un ensemble de contraintes matricielles. L'objectif est de déterminer une sous-matrice maximale cohérente, i.e., dont l'affectation des variables dans sa portée satisfait les contraintes et qui ne peut s'étendre en ligne en restant cohérente. Nous donnons une modélisation PPC du problème MMP et présentons différents types de contraintes matricielles. Nous étudions ensuite l'ordonnancement total ou partiel de patrons prélocalisés sur des séquences et permettant d'exclure des séquences prédéfinies. Nous présentons deux programmes par contraintes pour résoudre ces MMP ainsi qu'un algorithme génétique s'appuyant sur ces programmes pour passer l'échelle. Les résultats expérimentaux obtenus sur des jeux de séquences protéiques attestent de l'efficacité de l'approche.

Abstract

Itemset and pattern mining has numerous applications ranging from Marketing to Bioinformatics. We introduce a language, dubbed Maximal Matrix Problem (MMP), to model such problems. An instance of MMP is based on a matrix of finite domain variables and a set of matrix constraints. A solution is a maximal consistent submatrix whose assignment of the variables in its scope satisfies the constraints but cannot be extended over additional lines while preserving consistency. We propose a generic CP model for MMP and present various types of matrix contraints. We then tackle the problem of partially or totally ordering patterns that have been prelocalized over sequences in order to exclude predefined sequences. We present two CP models to solve these MMP together with a genetic algorithm. Experiments on datasets of protein sequences demonstrate the efficiency of the approach.

*Papier doctorant : Vincent Vigneron[1] est auteur principal.

1 Introduction

La programmation par contraintes (PPC) offre une alternative générique et efficace aux méthodes ad-hocs pour résoudre des problèmes de fouille de données. Différents modèles de PPC ont notamment été proposés pour la recherche d'itemset fréquent [3, 4] et de patrons [1, 10, 9]. Cet article propose un langage à base de contraintes, nommé Maximal Matrix Problem (MMP), pour modéliser et résoudre de tels problèmes. Une instance de MMP se définit par un domaine de matrices partiellement ordonné et un ensemble de contraintes matricielles. Une matrice résulte du choix d'une portée, prise sur un ensemble prédéfini de lignes (e.g., transactions, chaînes) et de colonnes (e.g., items, caractères), et du choix d'un coefficient pour chaque cellule de la portée, pris sur un domaine fini (e.g., absence/présence d'items, localisation de patrons).L'extension d'une matrice par adjonction de lignes ou de colonnes et choix de coefficients détermine un ordre partiel sur le domaine matriciel. L'objectif est de déterminer une matrice maximale cohérente.

Considérons la table en Figure 1 qui se compose d'un ensemble de patrons prélocalisés dans un jeu de séquences étiquetées positive ou négative. Une modélisation MMP consiste à associer les séquences aux lignes, les patrons aux colonnes et l'ensemble des localisations possibles au domaine de coefficients. Afin de discriminer séquences positives et négatives, on peut rechercher des patrons totalement ordonnés par leurs localisations sur les séquences positives mais dont l'ordre ou l'occurrence ne tient plus sur les séquences négatives. La matrice de portée $(\{s_1, s_2\}, \{\texttt{CC}, \texttt{DD}\})$ et coefficients $(s_1, \texttt{CC}) = 9$, $(s_1, \texttt{DD}) = 5$, $(s_2, \texttt{CC}) = 3$ et $(s_2, \texttt{DD}) = 1$ est solution du problème : ses coefficients sont conformes aux localisations autorisées ; elle couvre les lignes étiquetées positives ; ses colonnes satisfont l'ordre $\{\texttt{CC} > \texttt{DD}\}$; aucune combinaison de localisations autorisées pour CC et DD ne satisfait cet ordre sur s_3 ; et CC n'admet aucune localisation sur s_4.

Autrement dit, la matrice est cohérente avec les contraintes de base de données, de portée positive et d'ordre total, toute extension sur s_3 viole la contrainte d'ordre total et toute extension sur s_4 viole la contrainte de base de données. En ce sens, elle est cohérente et maximale en ligne. Elle l'est aussi en colonne, l'ordre ne pouvant être prolongé sur AA.

Nous restreignons le langage de contraintes matricielles aux contraintes de portée, de coefficients et de domaine. Les premières ne dépendent pas des choix de coefficients (e.g., un seuil de fréquence sur le nombre de lignes), les secondes ne dépendent pas des choix d'identifiants de lignes et de colonnes (e.g., égalité entre coefficients de toute portée) tandis que les contraintes de domaine restreignent les coefficients selon la portée mais indépendamment les uns des autres (e.g., compatibilité avec une base de données). Ces restrictions permettent de formuler la recherche de matrice cohérente comme un CSP et restent suffisamment souples pour modéliser des besoins pratiques en fouille d'itemsets ou de patrons. Le CSP se compose de variables représentant la portée et les coefficients de la matrice à déterminer, de contraintes d'indexation permettant d'anonymiser la portée, et d'une collection de contraintes modélisant chaque contrainte matricielle. L'encodage est linéaire en la taille maximum de matrice.

Établir la maximalité d'une matrice cohérente suppose de tester toute extension possible, c'est à dire, tout choix de coefficients sur tout sur-ensemble de lignes et de colonnes. Ce calcul est prohibitif dans le cas général puisqu'il suppose d'évaluer un nombre d'extensions doublement exponentiel. Il se simplifie si toute contrainte matricielle est monotone en ligne et en colonne (i.e., toute extension préserve la cohérence) ou bien anti-monotone (i.e., toute extension préserve l'incohérence). Sous cette hypothèse, une matrice cohérente est maximale si elle exclut chaque ligne et chaque colonne hors portée, i.e., si l'adjonction d'une ligne/colonne résulte en une matrice incohérente quel que soit le choix de coefficients des cellules adjointes. La preuve de maximalité peut ainsi se décomposer en excluant chaque ligne et chaque colonne séparément.

séquences	+/-	patrons		
		AA	CC	DD
s_1 = AACCDDAACCDD	+	{1,7}	{4,9}	{5,11}
s_2 = DDCCAADDCDD	+	{5}	{3}	{1,7,10}
s_3 = AACCAADDEDD	-	{1,5}	{3}	{7,10}
s_4 = DDAADDAA	-	{3,7}	$\emptyset$	{1,5}

FIGURE 1 – Une base de séquences constituée des positions de départ de différents patrons.

L'exclusion d'une ligne/colonne reste toutefois difficile puisque ce problème se ramène à la résolution d'un co-CSP : prouver que chaque combinaison de coefficients sur les cellules adjointes résulte en une extension violant une ou plusieurs contraintes anti-monotones. Nous proposons une résolution approchée en testant une condition d'*exclusion forte*. Il s'agit de déterminer si toutes les extensions satisfaisant les contraintes de domaine violent une même contrainte sur la ligne considérée. Chaque contrainte peut donc être traitée séparément et, en cas de succès, justifier à elle seule de l'exclusion de la ligne relativement à la base de données. Par exemple, la matrice présentée plus haut exclut fortement s_3 avec la contrainte d'ordre total et s_4 avec la base de données. A l'inverse, la matrice de portée $(\{s_1, s_2\}, \{\texttt{AA}, \texttt{DD}\})$ et de coefficients $(s_1, \texttt{AA}) = 1$, $(s_1, \texttt{DD}) = 5$, $(s_2, \texttt{AA}) = 5$, $(s_2, \texttt{DD}) = 10$ satisfait les contraintes d'ordre total $(\{\texttt{AA} < \texttt{DD}\})$ et d'écart minimum de 4 entre cellules de ligne. Elle ne s'étend pas sur s_4, les contraintes étant violées par toute extension. Pour autant, aucune des contraintes ne suffit à exclure s_4 à elle seule.

L'exclusion forte approxime donc l'exclusion totale mais elle est équivalente pour certains langages de contraintes matricielles. D'autre part, elle peut être de coût polynomial selon le langage. En outre, elle permet une algorithmique modulaire qui combine des propagateurs PPC dédiés à la cohérence avec des routines d'exclusion forte propres à chaque contrainte et appliquées à chaque ligne/colonne hors portée (routines appelées *co-propagateurs*). Enfin, les contraintes constituent des explications qui peuvent être mise à profit dans un contexte interactif où l'utilisateur affine son problème par ajout et retrait de contraintes, ou ré-étiquetage de lignes ou colonnes.

Le reste de l'article s'organise comme suit. La section 2 définit le problème MMP et donne une formulation PPC du calcul de matrice cohérente. La section 3 esquisse une modélisation MMP des problèmes de recherche d'itemsets et de patrons et présente quelques contraintes matricielles fondamentales. La section 4 se focalise sur la recherche de matrice de dimension bornée et sujette aux contraintes de base de données et de préservation de l'ordre requis, qu'il soit total ou maximum partiel. Nous montrons que ces problèmes sont NP-complet. Sont ensuite présentés des co-propagateurs pour les contraintes d'ordre total et d'ordre partiel maximum ainsi qu'un modèle complet et une approche génétique pour résoudre chaque problème. La section 5 présente une validation expérimentale sur des jeux de données biologiques qui confirment l'intérêt et les performances de l'approche. Enfin, la section 6 fait un rapide état de l'art et la section 7 présente quelques perspectives.

2 Maximal Matrix Problem

Nous formalisons dans cette section les notions de matrice, contrainte et extension sur un type matriciel et présentons le problème de décision MMP. On notera $[k]$ l'intervalle $\{j \in \mathbb{N} \mid 1 \leq j \leq k\}$ pour tout $k \in \mathbb{N}^+$, et $f(S)$ l'ensemble $\{f(u) \mid u \in S\}$ pour tout $f : A \to B$, $S \subseteq A$. Un type matriciel se définit par un ensemble ordonné d'identifiants de lignes et de colonnes et par un domaine fini pour les coefficients matriciels. L'ordre choisi est arbitraire et sans incidence sur les solutions d'une ins-

$$var\ 1..p : m; \quad (1)$$
$$array[1..p]\ of\ var\ 1..p : s1; \quad (2)$$
$$nvalue(s1, m); \quad (3)$$
$$increasing(s1); \quad (4)$$
$$s1[p] == s1[m]; \quad (5)$$
$$var\ 1..q : n; \quad (6)$$
$$array[1..q]\ of\ var\ 1..q : s2; \quad (7)$$
$$nvalue(s2, n); \quad (8)$$
$$increasing(s2); \quad (9)$$
$$s2[q] == s2[n]; \quad (10)$$
$$array[1..p, 1..q]\ of\ var\ D : v; \quad (11)$$

FIGURE 2 – Modélisation PPC de type matriciel.

$$among(p, s1, positives); \quad (1)$$
$$n \geq threshold; \quad (2)$$
$$\forall i = 1..p, j = 1..q,$$
$$(m == i/\backslash n == j) => allequal([v[k,l] | k = 1..i, l = 1..j]); \quad (3)$$
$$\forall i = 1..p, j = 1..q,\ v[i,j]\ in\ db[s1[i], s2[j]]; \quad (4)$$

FIGURE 3 – Modélisation PPC de contraintes matricielles.

tance de MMP. Sans perte de généralité, on supposera donc que les lignes et les colonnes sont numérotées (i.e., identifiées par des entiers) et triées en ordre croissant.

Définition 1 (Type matriciel)**.** *Un type matriciel M est un triplet $\langle p, q, D\rangle$ tel que $p \in \mathbb{N}^+$, $q \in \mathbb{N}^+$, et D est un ensemble fini. Les éléments de $[p]$, $[q]$ et D sont appelés, respectivement, lignes, colonnes, et coefficients de M.*

Une matrice se définit par le choix d'une portée et de coefficients sur cette portée relativement à un type donné. Ces choix caractérisent la dimension de la matrice en termes de nombre de lignes et de colonnes. Portée et coefficients se définissent par des fonctions d'indexation qui partagent les mêmes domaines de définition et assurent une représentation canonique de toute matrice.

Définition 2 (Matrice)**.** *Soient $M = \langle p, q, D\rangle$ un type matriciel, $m \in [p]$, et $n \in [q]$. Une matrice de type M et dimension (m, n) est une paire (s, v) telle que $s = (s_1, s_2)$, $s_1 : [m] \to [p]$, $s_2 : [n] \to [q]$, $v : [m] \times [n] \to D$ avec s_1 et s_2 strictement croissantes. s et v sont appelés portée et coefficients de x, respectivement.*

On notera $(s, v)_{m,n}$ pour indiquer la dimension (m, n) d'une matrice (s, v). L'ensemble des matrices de type M et taille (m, n) est dénoté $[\![M_{m,n}]\!]$ et l'ensemble des matrices de type M est dénoté $[\![M]\!] = \bigcup_{m\in[p],n\in[q]} [\![M_{m,n}]\!]$. La figure 2 présente une modélisation PPC de type matriciel. Les variables m (1) et n (6) représentent le nombre de lignes et de colonnes de la matrice à déterminer. Les tableaux de variables d'indices $s1$ (2) et $s2$ (7) représentent les fonctions d'indexation de portée. Les contraintes *nvalue* (3), *increasing* (4) et *élément* (5) assurent que $s1$ est strictement croissante sur $[m]$, les valeurs d'indice de ligne au delà du rang m étant forcées à celle du rang m. La modélisation est symétrique pour les colonnes (contraintes (8), (9) (10)). Le tableau de variables v (11) modélise les coefficients à déterminer, comme expliqué ci-après.

Une contrainte matricielle est une relation unaire sur le domaine matriciel qui se décompose en $p \times q$ relations. Chaque sous-relation est associée à une dimension de matrice (m, n) et utilisée de manière exclusive pour tester toute matrice (m, n) (condition (1) de la définition 3). Nous nous limitons à trois classes de contraintes : les contraintes de portée (condition (2)), les contraintes de coefficients (condition (3)), et les contraintes de domaine (condition (4)). La sémantique des contraintes de portée est indépendante du choix de coefficients. Celle des contraintes de coefficients est indépendante du choix de portée. Autrement dit, ces contraintes s'appliquent à des portées anonymisées. Les contraintes de domaine sont les seules contraintes matricielles liant portée et coefficients mais sont décomposables par restriction au niveau cellulaire.

Nous formalisons l'opération de restriction comme suit. Pour toute matrice $x = (s, v)_{m,n}$ et tout $I \subseteq s_1([m])$, $J \subseteq s_2([n])$, il existe $K = \{i_1, \ldots i_{|I|}\} \subseteq [m]$ et $L = \{j_1, \ldots j_{|J|}\} \subseteq [n]$ tels que $i_1 < i_2 < \ldots i_{|I|}$, $j_1 < j_2 < \cdots j_{|J|}$, $s_1(K) = I$ et $s_2(L) = J$. K et L sont uniques et on note $s_{1I} : [|I|] \to [p]$, $s_{2J} : [|J|] \to [q]$, $v_{IJ} : [|I|] \times [|J|] \to D$ définis par $s_{1I}(k) = s_1(i_k)$, $s_{2J}(l) = s_2(j_l)$ et $v_{IJ}(k, l) = v(i_k, j_l)$ $(k \in [|I|], l \in [|J|])$. $((s_{1I}, s_{2J}), v_{IJ})$ est la matrice, dénotée x_{IJ}, obtenue par restriction de x aux lignes de I et colonnes de J. On notera x_{ij} la matrice $x_{\{i\}\{j\}}$ pour tout $i \in s_1([m])$, $j \in s_2([n])$.

Définition 3 (Contrainte matricielle)**.** *Soit M un type matriciel. Une contrainte c de type M est une relation unaire sur $[\![M]\!]$ associée à un ensemble $\{c_{m,n} |\ c_{m,n} \subseteq [\![M_{mn}]\!], m \in [p], n \in [q]\}$ et vérifiant la condition*

$$\forall m \in [p], n \in [q], x \in [\![M_{m,n}]\!], c(x) \Leftrightarrow c_{m,n}(x) \quad (1)$$

et l'une au moins des conditions suivantes :

$$\forall (s, v), (s, v') \in [\![M]\!], c((s, v)) \Leftrightarrow c((s, v')) \quad (2)$$
$$\forall (s, v), (s', v) \in [\![M]\!], c((s, v)) \Leftrightarrow c((s', v)) \quad (3)$$
$$\forall x = (s, v)_{m,n} \in [\![M]\!], c(x) \Leftrightarrow \bigwedge_{\substack{i\in s_1([m]) \\ j\in s_2([n])}} c(x_{ij}) \quad (4)$$

La figure 3 illustre ces trois classes de contraintes. La contrainte de portée (1) impose que les lignes soient étiquetées positive par le biais de la contrainte $among$. La contrainte de portée (2) impose un seuil de fréquence en colonne. La contrainte de coefficients (3) impose l'égalité de tout coefficient dans la portée par le conditionnement de $p \times q$ contraintes $allequal$. Enfin, la contrainte de domaine (4) impose la compatibilité avec une base de données db encodée par un tableau $p \times q$ de sous-domaines de D (qui peuvent être vides ou égaux à D).

L'extension en ligne d'une matrice traduit l'adjonction d'une seule ligne à la portée avec choix de coefficients,

mais sans ajout de colonnes ni changement des coefficients de la matrice. L'extension détermine donc un ordre partiel sur le type matriciel.

Définition 4 (Extension matricielle)**.** *Soient* $x = (s, v)_{m,n}$ *et* $x' = (s', v')_{m',n'}$ *deux matrices de type* $\langle p, q, D\rangle$ *et* $i \in [p]$. x' *étend* x *sur* i*, dénoté* $x \sqsubset_i x'$*, si et seulement si* $\{i\} = s'_1([m']) \setminus s_1([m]) \wedge s'_2 = s_2 \wedge v'_{s_1([m])s_2([n])} = v$.

Une instance de MMP se définit par un type matriciel et un ensemble de contraintes scindé en contraintes de domaine, de coefficients et de portée. Comme leur nom l'indique, ces dernières sont utilisées pour controler la portée de matrice et ne sont donc pas prises en compte dans le calcul de maximalité, i.e., l'exclusion ne se jauge qu'avec les contraintes de domaine et de coefficients. Précisément, une solution de MMP est une matrice cohérente avec les contraintes et maximale pour l'exclusion forte relativement aux contraintes de domaine et de coefficients, i.e., une matrice qui satisfait les contraintes sur sa portée et dont l'extension en ligne viole systématiquement une même contrainte, la contrainte étant à déterminer pour chaque ligne. Étant donné un ensemble C de contraintes de type M et $x \in [\![M]\!]$, on utilisera $C(x)$ en lieu et place de $\bigwedge_{c\in C} c(x)$.

Définition 5 (MMP)**.** *Une instance de MMP est une paire* $\mu = \langle M, C\rangle$ *tel que* $M = \langle p, q, D\rangle$ *est un type matriciel et* $C = C_d \uplus C_s \uplus C_v$ *est un ensemble de contraintes de type* M *partitionné en contraintes de domaine* C_d*, de portée* C_s*, et de coefficients* C_v*.* μ *est satisfiable si et seulement si la proposition suivante est vérifiée :*

$$\exists x \in [\![M]\!], C(x) \wedge \bigwedge_{i\in[p]} \bigvee_{c\in C_v} \Big(\forall y \in [\![M]\!], (C_d(y) \wedge x \sqsubset_i y) \Rightarrow \neg c(y)\Big).$$

Une solution x de $\langle M, C\rangle$ est donc une matrice cohérente (i.e., satisfaisant C) et maximale (i.e. excluant fortement chaque ligne i hors portée par une contrainte de coefficients c spécifique à i). La figure 4 présente une modélisation PPC de la procédure d'exclusion forte dans le cas où, pour chaque contrainte de coefficients de l'instance, ce problème est polynomial. La procédure repose sur un un tableau de variables pseudo-booléennes r (1) où $r[i]$ vaut 1 si et seulement si la ligne i est dans la portée (contraintes (2) et (3)). Chacune de ces variables conditionne la tentative d'exclusion forte de la ligne si elle est hors portée. L'exclusion s'établit par une disjonction de contraintes d'exclusion (4). La contrainte d'exclusion $cop_k(i, db, s1, s2, v)$ est associée à la k-ième contrainte de coefficients, c, et a pour sémantique $(\forall y \in [\![M]\!], (d_B(y) \wedge x \sqsubset_i y) \Rightarrow \neg c(y))$, où x correspond à la matrice candidate $(m, n, s1, s2, v)$. On présente en section4 des co-propagateurs décidant ces contraintes pour les contraintes matricielles d'ordre.

$$array[1..p]\ of\ var\ 0..1 : r; \quad (1)$$
$$m = \sum_{i=1..p} r[i]; \quad (2)$$
$$\forall i = 1..p, r[s1[i]] == 1; \quad (3)$$
$$\forall i = 1..p, (r[i] == 0) => (cop_1(i, db, s1, s2, v)||..||cop_t(i, db, s1, s2, v)); \quad (4)$$

FIGURE 4 – Modélisation PPC de l'exclusion forte.

3 Modélisation et contraintes MMP

Nous présentons dans cette section quelques prédicats matriciels ainsi qu'une modélisation possible de problèmes de recherche d'itemsets et de patrons. La table 1 formalise les prédicats de base de données, de couverture, de cardinalité, de fréquence, d'ordre total et d'ordre partiel maximum. Chaque prédicat y est défini par sa signature (symbole et paramètres), sa sémantique et la sémantique du problème d'exclusion forte qui lui est associé. Le prédicat de base de données $\mathtt{d_B}$ est paramétré par une base de données B qui se définit par la matrice $p \times q$ des coefficients autorisés pour chaque cellule. Ce prédicat est anti-monotone pour l'extension $\sqsubset_i$, i.e., toute extension de matrice violant $\mathtt{d_B}$ viole également $\mathtt{d_B}$. Le prédicat de couverture $\mathtt{cover_R}$ est paramétré par un ensemble de lignes R et satisfait par toute matrice dont les lignes incluent R. Les prédicats de cardinalité $\mathtt{card}_{\leq \mathtt{w}}$ et de fréquence $\mathtt{freq}_{\leq \mathtt{w}}$ imposent qu'une matrice ait moins de w colonnes et w lignes, respectivement. Ces trois prédicats sont des prédicats de portée, indépendants des choix de coefficients.

Le prédicat $\mathtt{total}_<$ est paramétré par un ordre total strict $<$ sur D. $<$ induit sur chaque ligne de matrice un ordre partiel entre les colonnes qui dépend de la valeur des coefficients. $\mathtt{total}_<$ impose que ces ordres soient totaux et égaux sur toutes les lignes de la portée. Ce prédicat est clairement anti-monotone pour l'extension et exclure une ligne sur la base d'une matrice le satisfaisant revient à prouver l'insatisfiabilité d'un CSP binaire qui ne comporte que des contraintes d'inégalité $<$ (cf. table 1). Cette classe de CSP est polynomiale et nous présentons en section suivante un co-propagateur sous forme de CSP.

$\mathtt{partial}_{<,\alpha}$ est similaire à $\mathtt{total}_<$ mais se borne à calculer l'ordre partiel maximum qui est induit par $<$ entre les colonnes et commun à l'ensemble des lignes, ordre dénoté α en table 1. Cet ordre existe et est unique pour toute matrice donc $\mathtt{partial}_{<,\alpha}$ est toujours vérifiée. Cependant, nous utilisons ce prédicat pour l'exclusion en imposant qu'aucune extension de matrice cohérente ne préserve l'ordre partiel maximum qui lui est associé. Le problème d'exclusion reste polynomial et revient à montrer l'incohérence du CSP défini en table 1. Nous en donnons un co-propagateur en section suivante.

Nous présentons quelques modélisations MMP de problèmes de fouille utilisant ces prédicats. D'une part, la re-

TABLE 1 – Exemples de prédicats matriciels sur $M = \langle p, q, D \rangle$. Chaque ligne correspond à un prédicat c dont on donne la sémantique $c(x)$ pour $x = (s, v) \in [\![M]\!]$, et celle du problème d'exclusion d'une ligne i.

Prédicat	**Paramètres**	$c(x) \Leftrightarrow$	x exclut i avec c ssi
d_B	$B : [p] \times [q] \to 2^D$	$\bigwedge_{i \in [m], j \in [n]} (v(i,j) \in B(s_1(i), s_2(j)))$	$\bot$
$\texttt{cover}_{\texttt{R}}$	$R \subseteq [p]$	$R \subseteq s_1([m])$	N/A
$\texttt{card}_{\leq \texttt{w}}$	$w \in [q]$	$w \geq n$	N/A
$\texttt{freq}_{\leq \texttt{w}}$	$w \in [p]$	$w \leq m$	N/A
$\texttt{total}_{<}$	$<$ ordre total strict sur D	$\bigwedge_{j,k \in [n], j \neq k}$ $((\bigwedge_{i \in [m]} (v(i,j) < v(i,k)))$ $\vee (\bigwedge_{i \in [m]} (v(i,j) > v(i,k))))$	CSP (X, D, C) insatisfaisable avec $X = \{X_1, \ldots, X_n\}$ $X_j \in D_j,\ D_j = B(i, s_2(j))\ (j \in [n])$ $C = \{(X_j < X_k) \mid j, k \in [n]\,,\ v_{s_1(1)s_2(j)} < v_{s_1(1)s_2(k)}\}$
$\texttt{partial}_{<,\alpha}$	$<$ ordre total strict sur D	$\alpha : [n] \times [n] \to \mathbb{B}$ $\alpha(j,k) = \bigwedge_{i \in [m]} (v(i,j) < v(i,k))$	CSP (X, D, C) insatisfaisable avec $X = \{X_1, \ldots, X_n\}$ $X_j \in D_j,\ D_j = B(i, s_2(j))\ (j \in [n])$ $C = \{(X_j < X_k) \mid j, k \in [n]\,, \alpha(j,k)\}$

cherche d'itemsets fréquents parmi p items et q transactions (où chaque transaction est un sous-ensemble d'items) se modélise par le type matriciel $\langle p, q, \{1\} \rangle$ et une base de données B vérifiant $B(i,j) = D$ si l'item j appartient à la transaction i. Autrement dit, $B(i,j) = \emptyset$ si l'item j n'appartient pas à la transaction i. On garantit ainsi que seuls les itemsets présents dans une transaction satisfont la contrainte de domaine. On peut ensuite rechercher des itemsets fréquents en paramétrant le prédicat $\texttt{freq}_{\geq \texttt{w}}$.

Un ensemble de p chaînes de caractères sur lesquelles q patrons ont été prélocalisés (en calculant toutes les positions de départ possibles) peut se modéliser par un type matriciel $\langle p, q, D \rangle$ à domaine entier et une base de données consignant les localisations admissibles de chaque patron dans chaque séquence. On peut ainsi rechercher un ensemble commun de patrons et le contraindre par des prédicats de fréquence, cardinalité, et ordre partiel ou total. Dans le cas où les patrons correspondent à des sous-chaînes ou sous-séquences à trous constants, une matrice satisfiant le prédicat d'ordre total permet un alignement des chaînes couvertes. Lorsque les patrons correspondent aux lettres de l'alphabet utilisé, le prédicat d'ordre total détermine des sous-séquences communes sans répétition. L'usage supplémentaire d'un prédicat d'écart nul (non présenté ici) permet de calculer des sous-chaînes communes sans répétition. Enfin, l'association à chaque lettre d'un nombre de colonnes correspondant au nombre maximum d'occurrences de la lettre dans une chaîne permet de rechercher sous-chaîne et sous-séquence commune au sens général.

4 Calcul de matrices ordonnées

Dans cette section, nous abordons les prédicats d'ordre total et d'ordre partiel maximum. On étudie les problèmes dont toute solution est une matrice compatible avec une base de données (prédicat $\texttt{d}_{\texttt{B}}$), dont les lignes recouvrent un ensemble R prédéfini (prédicat $\texttt{cover}_{\texttt{R}}$) et dont les colonnes sont ordonnées, totalement (prédicat $\texttt{total}_{<}$) ou partiellement (prédicat $\texttt{partial}_{<,\alpha}$). Une matrice solution exclut donc toute ligne hors portée, soit par incompatibilité avec la base de données, soit par impossibilité de préserver l'ordre induit. On se restreint à rechercher parmi l'ensemble de solutions celles dont la fréquence et la cardinalité sont bornées supérieurement. Le problème MMP résultant est NP-complet dans ses variantes avec ordre total, avec ordre partiel maximum, ou sans contrainte d'ordre. Soit $\Gamma = \langle \Gamma_d, \Gamma_s, \Gamma_v \rangle$ où Γ_d, Γ_s et Γ_v sont des ensembles de prédicats de domaine, de portée, et de coefficients, respectivement. On note MMP$\langle \Gamma \rangle$ la classe d'instances MMP dont les contraintes de domaine, de portée et de coefficients instancient respectivement les prédicats de Γ_d, Γ_s et Γ_v sur le type matriciel choisi.

Théorème 1. *MMP$\langle \Gamma \rangle$ est NP-complet si* $\Gamma_d = \{\texttt{d}_{\texttt{B}}\}$, $\Gamma_s = \{\texttt{freq}_{\leq \texttt{k}}, \texttt{card}_{\leq 1}\}$ *et* $\Gamma_v \subseteq \{\texttt{total}_{<}, \texttt{partial}_{<,\alpha}\}$.

Proof. (1) On considère d'abord la classe d'instances sans contraintes d'ordre, i.e., $\Gamma_v = \emptyset$. Vérifier qu'une matrice est solution revient d'abord à tester si elle satisfait les contraintes de base de données, fréquence ($\leq k$) et cardinalité ($\leq l$). Ces tests s'effectuent clairement en temps polynomial. Vérifier qu'une matrice cohérente est maximale revient à tester l'exclusion de chaque ligne hors portée avec la base de données. Dans ce cas, il suffit de tester pour chaque ligne s'il existe une cellule incompatible avec la base pour pouvoir l'exclure. La vérification de solution est donc polynomiale et le problème est dans NP. Pour établir la NP-complétude, on réduit le problème de couverture par ensembles. Etant donné un entier l, un ensemble U fini, et un sous-ensemble S de l'ensemble des parties de U, il s'agit de déterminer l'existence d'un sous-ensemble T de S, de taille inférieure à l, et tel que l'union

des éléments présents dans les sous-ensembles de T est égal à U. La réduction consiste à définir le type matriciel $\langle k + |U|, |S|, D\rangle$, D quelconque, où l'on associe la colonne j à chaque élément s_j de S, la ligne i à chaque élément u_i de U ainsi que $k > 0$ lignes supplémentaires. La base de données autorise toute valeur pour toutes les cellules de ces k lignes. Elles sont donc nécessairement couvertes par toute matrice solution. Pour toute cellule (i, j) ($u_i \in U, s_j \in S$), on pose $B(i, j) = \emptyset$ ssi $u_i \in s_j$. On pose enfin les contraintes $\texttt{freq}_{\leq \texttt{k}}$ et $\texttt{card}_{\leq 1}$. Ainsi, une matrice maximale doit exclure chaque ligne u_i, i.e., une de ses colonnes correspond à un élément de S qui contient u_i. L'ensemble des couvertures solutions de taille $\leq l$ est donc en bijection avec l'ensemble des matrices solutions de ce MMP et la réduction s'effectue en temps polynomial. Le problème MMP$\langle\{\texttt{d}_\texttt{B}\}, \{\texttt{freq}_{\leq \texttt{k}}, \texttt{card}_{\leq 1}\}, \{\}\rangle$ est donc NP-complet. (2) On considère la classe d'instances avec contrainte d'ordre total, i.e., $\Gamma_v = \{\texttt{total}_<\}$. La vérification de matrice solution reste polynomiale car la cohérence d'une matrice avec la contrainte d'ordre total consiste à comparer des couples de coefficients et l'exclusion avec cette contrainte est aussi polynomiale (cf. table 1). Pour établir la NP-complétude, on considère la classe d'instances $\mu = \langle\langle p, q, D\rangle, C\rangle$ construites pour $\Gamma_v = \emptyset$ et vérifiant $D = [q]$ et $B(i, j) \subseteq \{j\}$ ($i \in [p]\,, j \in [q]$). Cette classe définit un problème NP-complet selon la preuve précédente. De plus, toute matrice cohérente avec la base de données satisfait nécessairement l'ordre total induit entre colonnes par la base, et cet ordre ne peut donc exclure aucune ligne hors portée. Cette classe se réduit donc à la classe d'instances avec contrainte d'ordre total qui est donc NP-complet. (3) Une preuve similaire s'établit pour la classe avec contrainte d'ordre partiel en utilisant cette fois un domaine singleton qui n'autorise donc que des ordres partiels maximaux vides. □

Nous présentons deux modèles PPC pour les classes de MMP avec contrainte d'ordre total et avec contrainte d'ordre partiel maximum. On se place dans le cadre de la recherche d'ensemble ordonné de patrons qui ont été prélocalisés sur un jeu de séquences étiquetées positive ou négative. La figure 7 présente deux exemples de solutions pour deux instances du problème d'ordre partiel. On suit la modélisation donnée en section 3 où les séquences, patrons et localisations sont assimilés, respectivement, aux lignes, colonnes et coefficients. En théorie, on cherche à calculer des matrices maximales et de portée bornée supérieurement en fréquence et cardinalité. Plutôt que d'explorer le front Pareto des solutions correspondant à cet objectif bi-critère, on recherche en priorité les solutions de fréquence minimum et, parmi ces solutions, celles de plus petite cardinalité. Puisque les séquences positives doivent être couvertes, on cherche donc à exclure le maximum de séquences négatives avec un nombre minimal de patrons.

Dans les deux modèles présentés, p et q désignent respectivement le nombre de séquences et le nombre de patrons. $\mathcal{P}^+$ et $\mathcal{P}^-$ sont deux sous-ensembles disjoints de $[p]$ représentant l'ensemble des séquences positives et l'ensemble des séquences négatives. $L[i, j]$ correspond à l'ensemble des localisations du patron j dans la séquence i. $L.row(i)$ correspond aux ensembles des localisations des patrons dans la séquence i.

$$\bigvee_{j\in[q]} c[j] \quad (1)$$

$$\forall i \in [p] : \quad sort\_perm(l.row(i), o, sl.row(s)) \quad (2)$$

$$\forall j \in [q] : \quad c[j] \leftrightarrow (o[j] \leq card) \quad (3)$$

$$\forall i \in [p] : \quad element(card, sl.row(i), last[i]) \quad (4)$$

$$\forall i \in \mathcal{P}^+ : \quad last[i] \notin \Gamma \quad (5)$$

$$\forall i \in \mathcal{P}^- \forall j \in [q] : \quad element(o[j], L\Gamma.row(i), sL\Gamma[i, j]) \quad (6)$$

$$\forall i \in \mathcal{P}^- : \quad minimum(sl[1], sL\Gamma[i, 1]) \quad (7)$$

$$\forall i \in \mathcal{P}^- \forall j \in [2..q] : \quad min\_at\_least(sl[j], sL\Gamma[i, j], sl[j-1]) \quad (8)$$

$$\forall i \in \mathcal{P}^- : \quad r[i] \leftrightarrow (last[i] \notin \Gamma) \quad (9)$$

$$minimize \quad q.\sum_{i\in\mathcal{P}^-} r[i] + card \quad (10)$$

FIGURE 5 – Modèle pour l'ordre total.

4.1 Un modèle PPC pour l'ordre total

La figure 5 présente le modèle PPC pour la résolution lexicographique du problème MMP$\langle\{\texttt{d}_\texttt{B}\}, \{\texttt{cover}_{\mathcal{P}^+}, \texttt{freq}_{\leq \texttt{k}}, \texttt{card}_{\leq 1}\}, \{\texttt{total}_<\}\rangle$.

Notations. $\epsilon = max_{i\in[p]}(|s_i|) + 1$ est une valeur strictement supérieure à l'ensemble des localisations possibles. $\Gamma = \{\epsilon, \ldots, \epsilon + [q]\}$ est un ensemble de valeurs qui permet de reproduire sur les séquences négatives tout ordre total commun aux séquences positives. Pour une séquence négative, avoir recours à des valeurs de Γ pour reproduire un ordre total signifie que cet ordre ne peut être reproduit avec les localisations autorisées par la base de données, donc la séquence ne supporte pas l'ordre total.

Variables. Pour tout $j \in [q]\,$, $c[j]$ est une variable booléenne qui dénote l'inclusion du patron j dans la solution. l est une matrice de variables entières telle que chaque cellule $l[i, j] \in \Gamma \cup L[i, j]$ correspond à la localisation choisie pour le patron j dans la séquence i. $o[j]$ est une variable entière qui correspond au rang du patron j dans l'ordre total. Pour tout $i \in \mathcal{P}^-$, $r[i]$ est une variable booléenne qui modélise l'exclusion de la séquence i, c'est-à-dire si l'ordre total ne peut pas être reproduit sur i. sl est une matrice d'entiers telle que chaque ligne $sl.row(i)$ correspond à la ligne $l.row(i)$ triée par ordre croissant. sL est une matrice de variables d'ensembles d'entiers telle que chaque ligne $sL.row(i)$ correspond à la ligne $L.row(i)$ permutée avec la fonction définie par o. $card$ est une variable entière correspondant au nombre de patrons sélectionnés. Enfin, $last[i]$ est une variable entière correspondant à la localisation du dernier patron de l'ordre total dans la séquence i.

Contraintes. L'équation (1) impose la sélection d'au

$$\bigvee_{j \in [q]} c[j] \quad (1)$$

$$\forall j, j' \in [q] : \quad a[j, j'] \leftrightarrow (c[j] \wedge c[j'] \wedge \bigwedge_{i \in \mathcal{P}+} (l[i, j] < l[i, j'])) \quad (2)$$

$$\forall i \in \mathcal{P}^- : \quad check\_partial\_order(c, a, L.row(i), r[i]) \quad (3)$$

$$minimise \quad q. \sum_{i \in \mathcal{P}^-} r[i] + \sum_{j \in [q]} c[j] \quad (4)$$

FIGURE 6 – Modèle pour l'ordre partiel.

moins un patron. L'équation (2) ordonne chaque séquence positive en utilisant la même permutation. [1] L'équation (3) impose que les rangs des patrons sélectionnés doivent précéder les rangs des autres patrons. L'équation (4) enregistre pour chaque séquence la localisation du dernier patron de l'ordre. L'équation (5) impose pour chaque séquence positive que la localisation du dernier patron sélectionné doit être valide, c'est-à-dire en dehors de Γ. Les équations (7) et (8) reproduisent, sur les séquences négatives, l'ordre total extrait des séquences positives. Cet ordre est reproduit en choisissant systématiquement les valeurs minimales envisageables. Si l'ordre total n'a pas pu être reproduit en sélectionnant les valeurs minimales alors il ne peut pas être reproduit avec les autres valeurs. L'équation (9) permet d'exclure les séquences négatives si la localisation du dernier patron de l'ordre total appartient à Γ. [2]

Objectif. L'objectif (10) est de maximiser le nombre de séquences exclues puis de minimiser le nombre de patrons sélectionnés.

4.2 Un modèle PPC pour l'ordre partiel

La figure 6 présente le modèle PPC pour la résolution lexicographique du problème MMP$\langle\{\mathtt{d_B}\}, \{\mathtt{cover}_{\mathcal{P}^+}, \mathtt{freq}_{\leq \mathtt{k}}, \mathtt{card}_{\leq 1}\}, \{\mathtt{partial}_{<,\alpha}\}\rangle$.

Variables. Pour tout $j \in [q]$, $c[j]$ est une variable booléenne qui réifie l'appartenance du patron j à la solution. l est une matrice de variables entières telle que chaque cellule $l[i, j] \in L[i, j]$ correspond à la localisation choisie pour le patron j dans la séquence i. a est une matrice de variables booléennes telle que chaque cellule $a[j, j']$ réifie la présence d'un arc de j vers j'. Pour tout $i \in \mathcal{P}^-$, $r[i]$ est une variable booléenne qui réifie l'exclusion de i.

Contraintes. L'équation (1) force la sélection d'au moins un patron dans l'ordre partiel. L'équation (2) impose que si j précède j', c'est-à-dire si $a[j, j'] = 1$, alors pour chaque séquence positive la localisation du patron j doit précéder la localisation du patron j' et réciproquement. L'équation (3) vérifie si l'ordre partiel est vérifié pour les séquences négatives. $check\_partial\_order$ effectue l'opération de vérification et affecte la valeur 0 a $r[i]$ si l'ordre partiel n'a pu être reconstitué. Si l'ordre partiel est totalement déterminé et que la séquence n'a pas pu être exclue alors r prend la valeur 1.

Objectif. L'objectif (4) est de maximiser le nombre de séquences exclues, puis de minimiser le nombre de patrons sélectionnés.

check_partial_order(c,a,D,r) est le co-propagateur utilisé dans l'équation (3). Il vérifie si une séquence peut satisfaire un ordre partiel. Une séquence est représentée par une liste d'ensembles (paramètre D) dont les éléments correspondent aux localisations des patrons dans la séquence. c est une liste de variables booléennes qui indique si un patron appartient à l'ordre partiel. a est une matrice de variables de booléennes qui représente la relation d'ordre entre les patrons. Le patron j précède le patron j' si $a[j, j'] = 1$. r est une variable booléenne qui indique si l'ordre partiel définit par le couple (c, r) peut être reproduit à partir des domaines fournis par D. Le co-propagateur utilise une valeur spéciale, noté $\epsilon = max(D) + 1$ et une liste de variables $starts$ telle que $\forall j \in [|c|]$ $starts[j] \in D[j] \cup \{\epsilon\}$. Les algorithmes 1 et 2 montrent la mise à jour des valeurs de $starts$ Le premier algorithme est appelé lorsque qu'un patron est sélectionné ou exclu tandis que le second est appelé lorsque un arc est sélectionné ou exclu. Le principe est le même que celui utilisé pour l'ordre total : l'ordre partiel va être reproduit en utilisant les plus petites localisations possibles de D (ligne 2 de l'algorithme 1, ligne 3 de l'algorithme 2). Si l'un des $starts$ prend la valeur ϵ alors la séquence ne peut pas reproduire l'ordre partiel (lignes 3-5 de l'algorithme 1, lignes 4-6 de l'algorithme 2). Si tout les c et a sont assignés sans qu'aucun $starts$ ne prenne la valeur ϵ alors la séquence satisfait l'ordre (lignes 8-10 de l'algorithme 1, lignes 9-11 de l'algorithme 2).

1 propagate_column(j)

```
1.  if c[j] = 1 then
2.      starts[j] ← min(D[j] ∪ {ε})
3.      if starts[j] = ε then
4.          r ← 0
5.          End of propagation
6.      for all j' ∈ [|c|] s.t. a[j,j'] = 1 do
7.          propagate_arc(j,j')
8.  if all_assigned(a) and all_assigned(c) then
9.      r ← 0
10.     End of propagation
```

2 propagate_arcs(j,j')

```
1.  if a[j,j'] = 1 and c[j]=1 and c[j'] = 1 then
2.      if starts[j'] ≥ starts[j] then
3.          starts[j'] ← min({l|(l ∈ D[j] ∪ {ε}) ∧ (l >starts[j])})
4.          if starts[j'] = ε then
5.              r ← 0
6.              End of propagation
7.          for all j'' ∈ [|c|] s.t. a[j',j''] = 1 do
8.              propagate_arc(j',j'')
9.  if all_assigned(a) and all_assigned(c) then
10.     r ← 0
11.     End of propagation
```

1. Dans la contrainte globale $sort\_perm(src, perm, dest)$, le vecteur $dest$ est la version triée du vecteur src et $perm$ est la permutation qui permet de passer de src à $dest$.

2. La contrainte globale $min\_at\_least(var, Set, min)$ signifie que var prend la plus petite valeur de Set strictement supérieure à min.

4.3 Approche génétique

Nous présentons ici un algorithme génétique pour MMP$\langle\{\mathtt{d_B}\}, \{\mathtt{cover}_{\mathcal{P}^+}, \mathtt{freq}_{\leq \mathtt{k}}, \mathtt{card}_{\leq \mathtt{l}}\}, \{\mathtt{partial}_{<,\alpha}\}\rangle$ où l'on cherche à exclure le maximum de séquences avec un nombre de patrons minimal. Cet algorithme (Algorithme 3) se base sur le modèle PPC de la section 4.2 qui est utilisé pour générer la population initiale mais également croiser et fusionner les solutions. Les paramètres de l'algorithme sont $SizeP$ - le nombre d'individus de la population -, $SizeT$ - le nombre d'individus sélectionnés pour le tournoi (le nombre de fils générés est $SizeT/2$) -, It - le nombre d'itérations effectuées -, et To - une limite de temps utilisée par le modèle PPC. Le paramétrage utilisé pour les expérimentations est abordé en section 5.2.

3 Genetic$(\mathcal{P}^+, \mathcal{P}^-, SizeP, SizeT, It, To)$

1. $pop \leftarrow \varnothing$
2. **for each** $i \in \{1..SizeP\}$ **do**
3. $\quad pop \leftarrow pop \cup solve(\mathcal{P}^+, \mathcal{P}^-, To)$
4. **for each** $i \in \{1..It\}$ **do**
5. $\quad tourn \leftarrow selectNPair(pop, SizeT/2)$
6. $\quad new \leftarrow \varnothing$
7. $\quad$ **for each** $(p1, p2) \in tourn$ **do**
8. $\quad\quad tmp \leftarrow merge(p1, p2)$
9. $\quad\quad new \leftarrow new \cup solve(\mathcal{P}^+, \mathcal{P}^-, tmp, To)$
10. $\quad devTourn \leftarrow \{p | (p, \_) \vee (\_, p) \in tourn\}$
11. $\quad selected \leftarrow selectBest(new \cup devTourn)$
12. $\quad pop \leftarrow (pop \backslash devTourn) \cup selected$
13. **return** $best(pop)$

L'algorithme crée une population d'individus générés aléatoirement (lignes 1-3). A chaque itération, il sélectionne n paires d'individus distincts (ligne 5) et crée un nouvel individu pour chaque paire d'individus (lignes 6-9). Sont conservés les n meilleurs individus parmi les individus du tournoi et ceux créés (lignes 10-12). Enfin, le meilleur individu de la population est retourné (ligne 13).

Les appels à `solve` aux lignes 3 et 9 sont des appels au modèle PPC. Tous les problèmes transmis à ce modèle sont résolus jusqu'à l'obtention de la solution optimale ou lorsque la limite de temps imposée au modèle est atteinte. Les deux appels à `solve` retournent un individu qui est représenté par un ensemble de patrons (le vecteur c du modèle PPC) et un ensemble d'arcs (la matrice a du modèle PPC) qui représentent à eux deux l'ordre partiel extrait. La stratégie de branchement adoptée pour obtenir des individus différents dans la population est la suivante : les patrons sont sélectionnés les uns après les autres avec une probabilité de $0,5$, puis les arcs sont sélectionnés en ordre aléatoire avec une probabilité de $0,5$. L'ordre de sélection aléatoire pour les arcs est essentiel car la relation d'ordre partiel est antisymétrique et sélectionner les arcs dans un ordre déterministe pourrait empêcher l'exploration de certaines solutions. L'appel à `merge` (ligne 8) crée un nouvel individu à partir de deux individus connus. Le nouvel individu est créé par l'intersection des deux individus parents. L'intersection est définie comme suit : si un patron est présent (resp., absent) dans les deux individus, alors le patron est présent (resp., absent) dans l'individu fils ; l'intersection pour les arcs suit une procédure identique. Les patrons ou arcs qui sont présents dans un parent mais pas dans l'autre ne sont pas contraints, et c'est sur ces patrons et arcs que le modèle PPC effectuera les branchements.

(a) LEAP_12_0

(b) SHSP_17_3

FIGURE 7 – Deux exemples d'ordres partiels extraits.

5 Expérimentations

Cette section présente les résultats obtenus sur deux bases de séquences protéiques : Late Embryogenesis Abundant Proteins (LEAP) et Small Heat Shock Proteins (SHSP). La bases LEAP [5] comporte 1371 protéines réparties en 13 classes par des biologistes et la base SHSP [6] contient 3765 protéines réparties en 25 classes. Les programmes testés ont été implémentés avec la version 4.4.0 de la bibliothèque Gecode et les expérimentations ont été effectuées sur un processeur 2-core Intel Core i5-3380M (2,90GHz) avec une limite de 8Go de mémoire vive. Ces expérimentations ont été réalisées sur des instances contenant des patrons extraits des deux bases avec un algorithme ad hoc [9]. Un patron est défini comme une séquence consécutive de caractères (acides aminés) sur l'alphabet enrichi d'un caractère joker. Un patron ne peut commencer ni se terminer par un joker. Par exemple, le patron `A.B` couvre la séquence `CAAB` mais ne couvre pas la séquence `CADDB` car un joker se substitue à un seul caractère. Une instance comporte un ensemble de séquences scindé en séquences positives $\mathcal{P}^+$ et séquences négatives $\mathcal{P}^-$, un ensemble de patrons et les localisations de chaque patron dans chaque séquence. Tout patron possède au moins une localisation dans chaque séquence positive. Chacune des classes a été traitée tout à tour comme étant la classe positive, les séquences des autres classes formant alors la classe négative. L'objectif des expérimentations est d'extraire un ordre partiel, constitué des patrons définis dans les instances, qui couvre intégralement les séquences positives et qui exclue un maximum de séquences négatives. On utilise le terme *motif* en référence à un ordre partiel sur des patrons.

5.1 Approche complète

Le tableau 2 présente les résultats du modèle PPC sur 7 des classes de LEAP et 17 classes de SHSP. La première colonne correspond au nom de l'instance (le premier nombre correspond à la classe, tandis que le second correspond au nombre maximum de jokers consécutifs que

TABLE 2 – Résultats obtenus sur les bases LEAP et SHSP.

	Inst.	$\lvert\mathcal{P}^+\rvert$	$\lvert\mathcal{P}^-\rvert$	Pat.	$\lvert\mathcal{P}^-_{\varnothing}\rvert$	$\lvert c\rvert$	$\lvert a\rvert$	T (s)
LEAP	1_10*	208	1163	20	94	11	26	600
	2_8*	92	1279	25	16	12	25	600
	3_0	34	1337	9	0	1	0	0
	5_2	66	1305	10	0	3	3	3
	10_8*	78	1293	21	54	18	30	600
	11_1	35	1336	19	0	5	6	125
	12_0	17	1354	5	0	2	1	0
SHSP	2_6	109	3656	14	0	8	10	221
	3_2	64	3701	13	0	3	3	3
	4_0	23	3742	10	0	3	3	1
	5_0	25	3740	12	0	2	1	2
	7_10	100	3665	11	105	10	13	62
	8_1	30	3735	15	0	5	5	33
	13_3	156	3609	11	0	5	5	3
	14_0	84	3681	20	0	1	0	4
	15_0	26	3739	20	0	1	0	5
	16_9	75	3690	6	0	4	3	0
	17_3	88	3677	9	0	5	6	2
	19_1	23	3742	24	0	4	6	55
	21_2	131	3634	6	0	3	3	0
	22_0	15	3750	36	0	2	1	38
	23_4	60	3705	6	1	5	6	1
	24_2	107	3658	7	0	3	3	1
	25_2	57	3708	16	0	5	7	15

peuvent contenir les patrons). Les colonnes suivantes correspondent dans l'ordre : au nombre de séquences positives, au nombre de séquences négatives, au nombre de patrons, au nombre de séquences négatives qui n'ont pas été exclues par le motif trouvé (0 signifie que toutes les séquences négatives ont été exclues par le motif), au nombre de patrons qui composent le motif, au nombre d'arcs que contient le motif et au temps d'exécution en secondes. Les instances étiquetées avec une étoile sont celles dont le modèle n'a pas pu extraire le motif optimal en moins de dix minutes. La figure 7 présente deux exemples de motifs extraits.

Les motifs extraits pour 4 des 7 classes de la base LEAP et 15 des 17 classes de SHSP permettent une exclusion totale de la classe négative. De plus, dans certains cas l'ordre partiel permet d'exclure des séquences qui n'auraient pas pu être exclues avec un ordre total ou par un ensemble non ordonné de patrons.

5.2 Approche génétique

Nous présentons ici les comparaisons, sur les trois instances les plus difficiles, de trois méthodes : l'approche complète (section 4), une approche aléatoire et l'approche génétique (section 4.3). Les trois approches se basent sur le même modèle PPC et les mêmes heuristiques de branchement que celles utilisées dans l'approche génétique. L'approche aléatoire s'arrête lorsqu'une solution a été trouvé tandis que les deux autres approches s'arrêtent après un délai de 15 minutes.

L'approche complète et l'approche génétique ont chacune été exécutées 100 fois sur chaque instance et l'approche aléatoire 1000 fois. L'approche génétique a été exécutée sur une population de cent individus sur 150 itérations avec 10 solutions sélectionnées pour le tournoi lors de chaque itération. Les solutions qui constituent la population ont été initialisées avec l'approche complète mais avec une limite de temps de une seconde par solution. L'opérateur utilisé pour améliorer les solutions obtenus par fusion est également le modèle PPC mais avec une limite de temps de une demi-seconde, sauf lorsque le nombre de variables de décisions libres était inférieur à 15. Dans ce cas précis, le modèle a été exécuté jusqu'à obtenir la solution optimale.

TABLE 3 – Exclusion moyenne.

		Complete			Random			Evolutionary		
	Inst.	Avg	Min	Stdev	Avg	Min	Stdev	Avg	Min	Stdev
LEAP	1_10	69.0	41	14.0	85.3	51	22.8	38.3	36	4.5
	2_8	55.1	5	39.9	53.9	7	41.6	5.0	5	0.1
	10_8	50.4	35	10.9	75.7	34	29.8	29.9	28	1.1

Pour les trois instances, l'approche génétique est celle qui a obtenu les meilleurs résultats en terme de qualité de solution et de stabilité. L'écart type sur les deux autres méthodes sont importants. Une fois que les patrons sont choisis le modèle PPC a tendance à explorer le même voisinage car ces patrons ne sont jamais remis en cause, par exemple en redémarrant la recherche à partir de zéro. L'approche génétique évite cet inconvénient car la population est initialisée aléatoirement et permet ainsi d'explorer plus de solutions de l'espace de recherche.

6 État de l'art

Plusieurs auteurs ont proposé des modèles génériques, basés sur la PPC ou SAT, pour formuler des problèmes connus de fouille de données. Ces approches permettent de formuler de façon déclarative et efficace de tels problèmes. Deux problèmes principaux ont été étudiés : la recherche d'itemsets fréquents et la recherche de sous-séquences communes.

Recherche d'Itemsets. Dans [3], les auteurs proposent une approche basée sur la PPC pour extraire des itemsets fréquents d'une base de transactions. Ce problème peut être représenté par une matrice de 0 et de 1, dont les lignes représentent les transactions et les colonnes les items associés aux transactions. Un 1 dans une cellule de la matrice indique que l'item associé à la colonne de la cellule appartient à la transaction associée à la ligne de la cellule. L'objectif est alors d'extraire des matrices contenant uniquement des 1. Un itemset est dit fermé si la matrice extraite ne peut pas être étendue en ligne ou en colonne sans introduire de 0. Dans [4], les auteurs formulent plusieurs problèmes liés à la recherche d'itemsets fréquents (k-term DNF, k-clustering,...).

Recherche de sous-séquences. Il existe principalement deux formes de séquences extraites par les algorithmes de fouille de données. Pour les distinguer, on parlera de sous-séquences et de patrons. Une sous-séquence est une séquence de caractères où seul l'ordre d'apparition des carac-

tères importe, la distance entre deux caractères de la sous-séquence n'étant pas discriminante dans la recherche. Un patron est une séquence de caractères consécutifs où la distance entre les caractères joue un rôle discriminant. Souvent les patrons incluent un caractère joker qui représente tous les caractères de l'alphabet sur lequel les séquences sont encodées. Il existe principalement deux catégories de méthodes : celles qui exposent les enracinements des caractères [8, 12] et les autres [7, 11]. La première catégorie est plus adaptée à la recherche de patrons, mais elle autorise également la recherche de sous-séquences. Dans [11] les auteurs proposent un modèle de programmation par contraintes pour extraire des sous-séquences ou des patrons. L'approche utilisée se base sur la contrainte *regular* [15] et un encodage des séquences d'entrées sous la forme d'automates finis déterministes. Dans [12], les auteurs proposent deux contraintes globales pour effectuer la tâche d'extraction de sous-séquences ou patrons. La première contrainte vérifie uniquement la présence d'enracinements, tandis que la seconde expose les enracinements ce qui autorise d'imposer des contraintes sur la distance entre les caractères et donc d'extraire des patrons en plus des sous-séquences. Dans [7], les auteurs proposent une méthode pour extraire des sous-séquences. Comme pour les approches ad hocs présentées dans [2, 13, 14], les auteurs introduisent une contrainte globale qui permet d'énumérer efficacement les sous-séquences fréquentes avec une recherche en profondeur d'abord (propre au solver PPC).

7 Conclusion

Nous avons présenté le problème de décision MMP pour traiter différentes tâches de fouilles de données. Ce problème consiste à déterminer une matrice cohérente avec un ensemble de contraintes matricielles et maximale en ligne à cet égard. Nous avons présenté deux variantes du problème qui ont des applications en bioinformatique : la recherche de patrons partiellement ou totalement ordonnés par leur localisations et permettant de discriminer des classes de séquences prédéfinies. Ces problèmes sont NP-complet et nous avons présenté deux modèles PPC ainsi qu'un algorithme génétique. Les résultats expérimentaux sur données biologiques sont prometteurs. Nous nous proposons par la suite de développer des co-propagateurs pour un spectre plus large de contraintes matricielles et également d'étudier différents algorithmes et stratégies de recherche afin d'améliorer le passage à l'échelle.

Références

[1] Emmanuel Coquery, Said Jabbour, Lakhdar Saïs, and Yakoub Salhi. A SAT-Based approach for discovering frequent, closed and maximal patterns in a sequence. In *ECAI*, pages 258–263, 2012.

[2] Christie I Ezeife and Yi Lu. Mining web log sequential patterns with position coded pre-order linked wap-tree. *DMKD*, 10(1) :5–38, 2005.

[3] Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining : A constraint programming perspective. *AI*, 175(12–13) :1951–1983, 2011.

[4] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-pattern set mining under constraints. *Trans. KDE*, 25(2) :402–418, 2013.

[5] Gilles Hunault and Emmanuel Jaspard. The late embryogenesis abundant proteins DataBase. http://forge.info.univ-angers.fr/~gh/Leadb/index.php, 2013.

[6] Gilles Hunault and Emmanuel Jaspard. The small heat shock proteins database. sHSPdb. http://forge.info.univ-angers.fr/~gh/Shspdb/index.php, 2013.

[7] Amina Kemmar, Samir Loudni, Yahia Lebbah, Patrice Boizumault, and Thierry Charnois. Prefix-projection global constraint for sequential pattern mining. *arXiv preprint arXiv :1504.07877*, 2015.

[8] Amina Kemmar, Willy Ugarte, Samir Loudni, Thierry Charnois, Yahia Lebbah, Patrice Boizumault, and Bruno Cremilleux. Mining relevant sequence patterns with cp-based framework. In *ICTAI*, pages 552–559. IEEE, 2014.

[9] David Lesaint, Deepak Mehta, Barry O'Sullivan, and Vincent Vigneron. A Decomposition Approach for Discovering Discriminative Motifs in a Sequence Database. In *ECAI*, Prague, Czech Republic, 2014.

[10] J.-P. Métivier, S. Loudni, and T. Charnois. A Constraint Programming Approach for Mining Sequential Patterns in a Sequence Database. In *LML'13*, pages 1–15, 2013.

[11] Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. A constraint language for declarative pattern discovery. In *ACM*, SAC '12, page 119–125, New York, NY, USA, 2012. ACM.

[12] Benjamin Negrevergne and Tias Guns. Constraint-based sequence mining using constraint programming. *arXiv preprint arXiv :1501.01178*, 2015.

[13] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, pages 0215–0215. IEEE, 2001.

[14] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. Mining access patterns efficiently from web logs. In *KDDM. Current Issues and New Applications*, pages 396–407. Springer, 2000.

[15] Gilles Pesant. A regular language membership constraint for finite sequences of variables. In *CP*, pages 482–495. Springer, 2004.