Une Approche par Décomposition pour la Découverte de Motifs sur Données Séquentielles

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

Une Approche par Décomposition pour la Découverte de Motifs Discriminants sur Données Séquentielles

Vincent Vigneron[1] **David Lesaint**[1] **Deepak Mehta**[2] **Barry O'Sullivan**[2]

[1] LERIA, Université d'Angers, France

[2] INSIGHT Centre for Data Analytics Department of Computer Science, University College Cork, Ireland

{david.lesaint,vincent.vigneron}@info.univ-angers.fr {d.metha,b.osullivan}@4c.ucc.ie

Résumé

La Programmation par Contraintes est une approche récente en Fouille de Données qui cherche à pallier au manque de flexibilité des algorithmes dédiés. Cet article s'inscrit dans cette démarche et traite de la recherche de motifs discriminants dans une base de séquences étiquetées positives ou négatives. Nous définissons un motif comme un ensemble de c-blocs où un c-bloc correspond à l'enracinement d'une sous-séquence commune aux séquences positives et dont le nombre de caractères libres entre deux sous-chaînes reste constant d'une séquence à l'autre. Nous reformulons des contraintes classiques de fouille pour ce modèle de motif et présentons de nouvelles contraintes qui éliminent des motifs redondants ou répondent à des besoins applicatifs. Nous modélisons le calcul de motif discriminant comme un problème d'optimisation paramétré par deux ensembles de contraintes dites positives et négatives. Un motif solution doit satisfaire toutes ces contraintes sur la classe positive et exclure le plus grand nombre de séquences négatives, c'est-à-dire, toute séquence où l'on ne peut l'étendre sans violer de contraintes négatives. Nous nous focalisons sur la classe de problèmes visant à calculer des motifs fermés, sans réplications et couvrant minimalement la classe négative. Nous présentons une méthode qui construit des c-blocs fermés de manière ad-hoc puis détermine un motif solution par résolution d'un problème de couverture minimum par ensembles. Des résultats expérimentaux obtenus sur des bases de séquences protéiques en démontrent l'efficacité.

1 Introduction

La fouille de données revêt une importance majeure dans de nombreux domaines allant de la Bio-informatique à la Cyber-sécurité. Elle vise à extraire des caractéristiques propres à un ensemble de données, par exemple, la présence d'itemsets fréquents dans une base de transactions. De nombreux algorithmes de fouille ont été conçus selon le type de données à traiter (itemsets, séquences, données étiquetées ou non, etc.) ou le calcul à mener (apprentissage de concept, partitionnement, etc). Les algorithmes dédiés étant difficilement adaptables lorsque les besoins évoluent, des approches fondées sur la Programmation par Contraintes (PPC) ont été proposées pour une formulation plus déclarative des problèmes de fouille [3, 4, 9], l'objectif étant de concilier flexibilité de modélisation et efficacité de calcul.

Cet article s'inscrit dans cette démarche et aborde le problème de la recherche de motifs discriminants dans une base de séquences prépartitionnée en classes. Ce problème est d'intérêt en Bio-informatique pour valider a posteriori le partitionnement expert d'un jeu de séquences protéiques en signant chaque classe par un motif exclusif. Sans perte de généralité, nous considérons une base bi-partitionnée en *séquences positives* et *séquences négatives*. L'objectif est alors d'identifier un motif commun et exclusif à la classe des séquences positives. Nous définissons un motif comme un ensemble de *c-blocs* (blocs de classe). Chaque c-bloc correspond à l'enracinement d'une sous-séquence commune aux séquences positives et dont le nombre de caractères libres entre deux sous-chaînes est invariant d'une séquence à l'autre.

Ce modèle de motif permet de distinguer contraintes sur

c-blocs (par exemple, expressions régulières) et contraintes entre c-blocs (par exemple, absence de répétitions) sans préjuger d'un traitement unifié ou séparé. Nous reformulons d'une part les contraintes classiques de fermeture et couverture. Ces contraintes ont été formalisées dans des approches PPC pour d'autres types de calcul : motifs unaires [3] et n-aires [4] à base d'itemsets, motifs unaires pour séquence simple [1, 2] ou séquences multiples [9]. Nous présentons d'autre part des contraintes de non-inclusion et de non-réplication entre c-blocs. Ces dernières visent à éliminer certains motifs redondants à l'instar des contraintes de fermeture et de couverture.

Nous modélisons la recherche de motif discriminant sous la forme d'un problème d'optimisation sous contraintes (Motif Discovery Problem) paramétré par deux ensembles de contraintes dites positives et négatives. Un motif solution doit satisfaire toutes ces contraintes sur la classe positive et exclure le plus grand nombre de séquences négatives. L'exclusivité se mesure au nombre de séquences négatives sur lesquelles on ne peut étendre le motif sans violer l'une des contraintes négatives. Un MDP tolère des critères additionnels basés sur des mesures telle que la *variabilité* de motif (nombre maximum de caractères libres successifs au sein des c-blocs).

Le cadre MDP laisse toute flexibilité dans le choix et la répartition des contraintes à traiter. Les propriétés de contraintes (anti-monotonie, subsomption, etc) peuvent être exploitées à cet égard. Par exemple, toute extension sur la classe négative d'un motif sans réplications étant nécéssairement sans réplications, il est inutile de considérer la contrainte de non-réplication pour la preuve d'exclusivité. De même, tout motif sans réplications étant nécéssairement sans inclusions, cette dernière contrainte peut être ignorée si la non-réplication est requise.

Nous nous focalisons sur la classe de problèmes MDP qui vise au calcul de motifs fermés sur la classe positive, sans réplications et couvrant minimalement la classe négative (Replication-free MDP). Nous montrons que, sous certaines conditions portant sur la fonction objectif, tout RMDP satisfaisable admet un motif solution constitué de c-blocs maximaux pour la relation de réplication. Pour le cas général, nous présentons une méthode de résolution en deux étapes qui consiste à générer un ensemble de c-blocs fermés et sans inclusions puis à déterminer un motif solution. Nous considérons précisément le cas où la fonction objectif combine les critères suivants par ordre de priorité : exclusion maximum, variabilité minimum et cardinalité minimum. La méthode utilise un algorithme ad-hoc pour le calcul de c-blocs et ramène la seconde étape à la résolution d'un problème de couverture minimum par ensembles.

Des résultats expérimentaux obtenus sur deux bases de séquences protéiques [6, 7, 8] attestent de son efficacité. Ces résultats permettent par ailleurs d'exhiber des motifs non séquentialisables, autrement dit, des motifs incalculables à l'aide d'algorithmes d'alignement de séquences. En ce sens, ils légitiment le cadre MDP qui laisse à l'utilisateur le libre choix des contraintes à satisfaire selon les résultats obtenus à chaque étape d'un processus de fouille.

Le reste de l'article s'organise comme suit. La section 2 présente le modèle de motif à base de c-blocs et formalise les problèmes MDP et RMDP. La section 3 décrit la méthode de résolution de RMDP. La section 4 présente les résultats expérimentaux et en fournit une analyse. La section 5 conclut.

2 Définitions

Soit Σ un alphabet composé d'un nombre fini de caractères dits solides, une séquence sur Σ est définie par une suite de caractères $s_1, \ldots, s_n \in \Sigma^n$ ($n \in \mathbb{N}^*$). La longueur d'une séquence s est notée $|s|$. Une base de séquences bipartitionnée D est l'union de deux ensembles disjoints D^+ et D^- de séquences sur Σ ($D \subseteq \Sigma^*$ et $D = D^+ \cup D^-$). $D+$ (respectivement, D^-) est la classe des séquences dites positives (resp., négatives). Un hash est un caractère additionnel $\#$ qui n'appartient pas à Σ ($\# \notin \Sigma$). Un patron est une séquence sur $\Sigma \cup \{\#\}$ qui démarre et se termine par un caractère solide et est de longueur supérieure ou égale à 2. Les définitions qui suivent font référence à une base de séquences D sur un alphabet Σ. On notera $[n]$ l'intervalle $\{i \mid 1 \leqslant i \leqslant n\}$ pour $n \in \mathbb{N}$ et $[t]$ l'ensemble $[|t|]$ pour toute séquence, patron ou ensemble t.

Définition 1 (Patron) *Un patron est une séquence $p \in \Sigma.(\Sigma \cup \{\#\})^*.\Sigma$. On note p_i le $i^{ème}$ caractère de p pour tout $i \in [p]$, $\pi(p) = \{k \mid p_k \neq \#\}$ l'ensemble des positions des caractères solides dans p et $\pi(p)(i)$ la position du $i^{ème}$ caractère solide de p dans p pour tout $i \in [\pi(p)]$.*

Un patron peut apparaître plus d'une fois dans une même séquence. Par exemple, le patron `A##C#C` est enraciné aux positions 1 et 3 dans la séquence `ADACDCEC`. Un patron peut également apparaître dans différentes séquences de la base de données et l'on appelle couverture d'un patron l'ensemble des séquences dans lesquelles il est enraciné. Un patron est dit positif si sa couverture inclut la classe positive.

Définition 2 (Enracinement et couverture d'un patron) *Un patron p est enraciné dans une séquence s à la position $l \in [s]$, noté $p \subseteq_l s$, si $p_k = s_{l+k-1}$ ou $p_k = \#$ pour tout $k \in [p]$. La couverture d'un patron p, notée $\varphi(p)$, est l'ensemble $\varphi(p) = \{s \in D \mid \exists l \in [s],\ p \subseteq_l s\}$.*

Un bloc correspond à l'enracinement d'un patron dans une séquence. Un c-bloc correspond à l'enracinement d'un patron positif sur toute sa couverture. Un c-bloc résulte

donc du choix d'un bloc par séquence figurant dans la couverture du patron. Un motif est un ensemble de c-blocs et l'on définit sa couverture comme étant l'intersection des couvertures de ses c-blocs. Par définition, tout motif couvre la classe positive.

Définition 3 (Bloc, c-bloc et motif) *Un bloc est un triplet (p, l, s) où s est une séquence de D et p un patron enraciné à la position l dans s ($p \subseteq_l s$). Un c-bloc est une paire (p, δ) où p est un patron positif et δ est une fonction qui enracine p sur sa couverture : $D^+ \subseteq \varphi(p)$, $\delta \in (\varphi(p) \rightarrow \mathbb{N}^*)$ et $(p, \delta(s), s)$ est un bloc pour tout $s \in \varphi(p)$. La couverture d'un c-bloc $c = (p, \delta)$ est la couverture de son patron : $\varphi(c) = \varphi((p, \delta))$. Un motif est un ensemble de c-blocs. La couverture d'un motif Π est l'ensemble $\varphi(\Pi) = \bigcap_{c \in \Pi} \varphi(c)$.*

La Figure 1 illustre patrons et blocs pour une base de trois séquences. Par exemple, AA est un patron positif ; $(\texttt{AA}, 6, s_1)$, $(\texttt{AA}, 8, s_2)$ et $(\texttt{AA}, 1, s_3)$ sont des blocs pour ce patron et $(\texttt{AA}, \{6, 8, 1\})$ est donc un c-bloc possible[1]. $(\texttt{AAC\#E}, \{1, 1\})$ est un autre exemple de c-bloc. A l'inverse, $(\texttt{YE}, \{4\})$ n'est pas un c-bloc car le patron YE n'est pas positif. $\{(\texttt{A\#C}, \{1, 1, 1\}), (\texttt{AAC}, \{1, 1, 1\}), (\texttt{AAC\#E}, \{1, 1\}), (\texttt{AA}, \{6, 8, 1\})\}$ est un exemple de motif dont la couverture est D^+.

Nous recherchons des motifs qui sont communs et exclusifs à la classe positive, c'est-à-dire qui satisfont les contraintes prescrites sur la classe positive, et qui violent au moins une contrainte lorsqu'ils sont étendues aux séquences négatives. Chaque contrainte sur motif s'interprète donc relativement au choix d'un ensemble de séquences selon le calcul : classe positive pour le calcul de motif commun et sur-ensemble de la classe positive pour la preuve d'exclusivité.

Définition 4 (Contrainte sur motifs) *Soit C l'ensemble des c-blocs pour une base D. Une contrainte r sur motifs de D est une fonction $r : 2^D \times 2^C \rightarrow \{false, true\}$. On dit qu'un motif $\Pi \subseteq C$ satisfait une contrainte r sur $X \subseteq D$ si $r(X, \Pi) = true$.*

	s_1	s_2	s_3
Séquences	AACYEAA	AACZEZZAA	AAC
Blocs	A#C	A#C	A#C
	AAC	AAC	AAC
	AAC#E	AAC#E	
	AA	AA	AA

FIGURE 1 – Exemple de blocs pour une base de séquences $D^+ = \{s_1, s_2\}$ et $D^- = \{s_3\}$.

1. Nous supposons un ordre sur les séquences de D et nous représentons la fonction d'enracinement d'un c-bloc par son image ordonnée par rapport à D.

Nous présentons ci-dessous les contraintes de non-inclusion, non-réplication, couverture et fermeture. Les contraintes de non-inclusion et de non-réplication portent sur des couples de c-blocs alors que les contraintes de couverture et de fermeture portent sur un seul c-bloc à la fois. Ces contraintes permettent d'éliminer des motifs redondants en se basant sur les patrons ou les blocs de c-blocs.

La réplication est une contrainte de subsomption sur les patrons. Précisément, un patron p réplique un patron p' si p est obtenu à partir de p' en y remplaçant des hashs par des caractères solides ou en l'étendant à gauche ou à droite avec des caractères solides ou des hashs. Par exemple, le patron AAC#E réplique AAC qui lui même réplique A#C. L'inclusion est une contrainte de subsomption sur les blocs. Précisément, un bloc b inclut un bloc b' si b est obtenu à partir de b' en y remplaçant des hashs par des caractères solides ou en ajoutant des caractères solides ou des hashs à gauche ou à droite. Par exemple, le bloc $(\texttt{AAC\#E}, 1, s_1)$ inclut $(\texttt{AAC}, 1, s_1)$ qui lui même inclut $(\texttt{A\#C}, 1, s_1)$ en Figure 1.

Définition 5 (Inclusion et réplication de blocs) *Soit $b = (p, l, s)$ et $b' = (p', l', s)$ deux blocs pour une séquence $s \in D$. b inclut b', noté $b \sqsupseteq b'$, si $\{k + l \mid k \in \pi(p)\} \supseteq \{k + l' \mid k \in \pi(p')\}$. b réplique b', noté $b \succeq b'$, si $\exists b'' = (p, l'', s)$, $b'' \sqsupseteq b'$.*

Il y a réplication entre deux c-blocs s'il y a réplication entre leurs patrons. Il y a inclusion entre deux c-blocs sur un ensemble de séquences $X \subseteq D$ si ces c-blocs ont une séquence commune dans X sur laquelle l'inclusion se vérifie entre leurs blocs.

Définition 6 (Inclusion et réplication de c-blocs) *Soit $c = (p, \delta)$ et $c' = (p', \delta')$ deux c-blocs. c réplique c', noté $c \succeq c'$, si $\exists s \in D^+$, $(p, \delta(s), s) \succeq (p', \delta'(s), s)$. c inclut c' sur $X \subseteq D$, noté $c \sqsupseteq_X c'$, si $\exists s \in X \cap \varphi(c) \cap \varphi(c')$, $(p, \delta(s), s) \sqsupseteq (p', \delta'(s), s)$.*

La contrainte de couverture d'un ensemble $X \subseteq D$ par un motif stipule que la couverture du motif inclue X. La contrainte de fermeture stipule que les c-blocs du motif ne puissent pas être étendus, c'est à dire, ne puissent pas être répliqués tout en préservant leurs enracinements. Précisément, un c-bloc est fermé sur $X \subseteq D$ s'il n'existe aucun c-bloc l'incluant strictement sur toute séquence commune dans X. Par exemple, le c-bloc $(\texttt{AAC}, \{1, 1, 1\})$ en Figure 1 n'est pas fermé sur D^+ car $(\texttt{AAC\#E}, \{1, 1, 1\})$ l'inclut strictement sur toute séquence positive mais il est fermé sur D.

Définition 7 (Fermeture de c-blocs) *Un c-bloc (p, δ) est fermé sur $X \subseteq D$ s'il n'existe pas de c-bloc (p', δ') tel que $(p', \delta'(s), s) \sqsupsetneq (p, \delta(s), s)$ pour tout $s \in X \cap \varphi((p, \delta)) \cap \varphi((p', \delta'))$.*

Nous formalisons maintenant ces contraintes sur motifs relativement à un ensemble de séquences $X \subseteq D$. Un motif couvre X si sa couverture inclut X. Un motif est fermé sur X si ses c-blocs sont fermés sur X. Un motif est sans-réplications sur X si il n'y a pas de réplication entre ses c-blocs. Un motif est sans-inclusions sur X s'il n'y pas d'inclusion entre ses c-blocs sur X. Formellement [2],

$$couverture_X(\Pi) \Leftrightarrow X \subseteq \varphi(\Pi)$$
$$fermeture_X(\Pi) \Leftrightarrow \forall c \in \Pi,\ c \textit{ est fermé sur } X$$
$$non-replication_X(\Pi) \Leftrightarrow \forall c, c' \in \Pi,\ c \geq c' \Rightarrow c = c'$$
$$non-inclusion_X(\Pi) \Leftrightarrow \forall c, c' \in \Pi,\ c \sqsupseteq_X c' \Rightarrow c = c'$$

Le motif $\{(\texttt{AAC}, \{1,1,1\}), (\texttt{AAC\#E}, \{1,1\})\}$ en Figure 1 couvre D^+ mais ne couvre pas D et il n'est ni fermé, ni sans-inclusions sur sa couverture. Le motif $\{(\texttt{AAC}, \{1,1,1\}), (\texttt{AA}, \{6,8,1\})\}$ couvre D et est sans-inclusions mais il n'est ni fermé, ni sans-réplications sur D^+. Le motif $\{(\texttt{AAC\#E}, \{1,1\}), (\texttt{AA}, \{6,8,1\})\}$ couvre D^+ et il est fermé et sans-inclusions sur sa couverture mais il n'est pas sans-réplications.

D'autres contraintes peuvent être envisagées : contraintes sur patrons (par exemple, restreindre les caractères solides admissibles, borner le nombre de hashs, et plus généralement imposer des contraintes sous la forme d'expressions régulières [9]), contraintes sur c-blocs (par exemple, classes de substitutions autorisées pour tout hash), ou contraintes entre c-blocs (par exemple, contraintes sur le nombre de réplications ou de répétitions de c-blocs).

Le fait d'opérer sur des données séquentielles permet aussi de contraindre le positionnement ou l'ordonnancement interne de motif. En effet, chaque bloc correspond à un intervalle de positions et des relations tirées d'algèbres de points ou d'intervalles peuvent être utilisées pour imposer un ordonnancement cohérent des c-blocs d'un motif sur sa couverture. On peut notamment rechercher des motifs "séquentiels" en imposant un ordre total entre c-blocs par le biais de contraintes de précédence.

Ce calcul est une approche possible (non développée ici) au problème d'alignement de séquences multiples en Bio-informatique [5]. Un alignement s'obtient par une série de substitutions, suppressions ou insertions de caractères sur chaque séquence. Or tout c-bloc détermine un alignement possible des séquences positives, chaque caractère libre pouvant s'interpréter comme un point de substitution (par exemple, un point de mutation évolutionnaire dans une séquence protéique). D'autre part, toute variation d'écart entre c-blocs successifs d'un motif sur différentes séquences peut s'interpréter comme une série de suppressions ou d'insertions (par exemple, suppressions d'acides aminés dans une séquence protéique).

2. Pour une contrainte sur motif r, nous notons $r_X(\Pi)$ pour signifier $r(X, \Pi) = true$ et $\neg r_X(\Pi)$ pour signifier $r(X, \Pi) = false$.

Nous présentons maintenant le cadre MDP pour le problème de recherche de motifs discriminants. Un MDP est paramétré par un ensemble de contraintes et requiert qu'un motif solution satisfasse toutes ces contraintes sur la classe positive et qu'il exclue le plus grand nombre de séquences négatives. Un motif exclut une séquence négative si on ne peut "l'étendre" à cette séquence sans violer l'une des contraintes. Autrement dit, tout motif identique sur la classe positive viole nécessairement une contrainte lorsque l'interprétation incorpore la séquence négative.

Définition 8 (Mesure d'exclusion) *Soient Π et Π' deux motifs, on note $\Pi =_+ \Pi'$ s'il existe une bijection $f : \Pi \rightarrow \Pi'$ vérifiant $\forall (p, \delta) \in \Pi,\ f((p, \delta)) = (p', \delta') \Rightarrow (p = p' \wedge \delta_{|D^+} = \delta'_{|D^+})$. Soit R un ensemble de contraintes sur motifs, $exc_R(\Pi) = |\{s \in D^- \mid \forall \Pi' =_+ \Pi, \exists r \in R, \neg r(\varphi(\Pi) \cup \{s\}, \Pi')\}|$ est la mesure d'exclusion de Π et $inc_R(\Pi) = |D^-| - exc_R(\Pi)$ est sa mesure d'inclusion.*

Pour plus de flexibilité, un MDP permet de limiter l'ensemble des contraintes à considérer pour la measure d'exclusion. En effet, les contraintes monotones sur ensembles de séquences ou dont la sémantique ne dépend pas de la couverture de motif peuvent être écartées sans risques. C'est le cas la contrainte de non-réplication qui est indépendante de la couverture de motif et de la contrainte de fermeture qui est monotone (un motif fermé sur X l'est pour tout $Y \supseteq X$). A l'inverse, les contraintes de couverture et de non-inclusion sont anti-monotones et cette dernière sous-somme la la contrainte de non-réplication.

Définition 9 (MDP) *Un problème de découverte de motifs est un n-uplet $P = (D, R^+, R^-, <)$ où $D = D^+ \uplus D^-$ est une base de séquences bi-partitionnée, R^+ et R^- sont des ensembles (potentiellement vides) de contraintes sur motifs, et $<$ est un pré-ordre strict [3] sur les motifs vérifiant $\Pi < \Pi' \Rightarrow inc_{R^-}(\Pi) \leqslant inc_{R^-}(\Pi')$. Une solution de P est un motif minimal pour $<$ qui satisfait toutes les contraintes de $R^+ \cup R^-$ sur D^+.*

Notons qu'un MDP tolère l'utilisation de critères d'optimisation secondaires. Nous présentons ci-dessous deux mesures - cardinalité et variabilité de motif - qui peuvent être combinées avec la mesure d'inclusion. La variabilité d'un motif est le nombre maximum de hashs consécutifs des patrons de ses c-blocs.

Définition 10 (Mesures) *Soit Π un motif, $card(\Pi) = |\Pi|$ est la cardinalité de Π, et $slack(\Pi) = \max_{(p,\delta)\in\Pi, i\in[|\pi(p)|-1]} (\pi(p)(i+1) - \pi(p)(i) - 1))$ est la variabilité de Π.*

3. Un pré-ordre strict est un ordre partiel strict pour lequel la relation d'incomparabilité associée est transitive.

Nous définissons ci-dessous la classe RMDP (Replication-free MDP) de MDP dont les motifs solutions sont fermés sur la classe positive, sans-réplications et couvrant minimalement la classe négative.

Définition 11 (RMDP) *RMDP est la classe de MDP telle que* $R^+ = \{\texttt{fermeture}, \texttt{non-réplication}\}$ *et* $R^- = \{\texttt{couverture}\}$.

De par les propriétés de la contrainte de non-réplication, notons que tout RMDP $(D, \{\texttt{fermeture}, \texttt{non-réplication}\}, \{\texttt{couverture}\}, <$ $)$ a mêmes solutions que le MDP $(D, \{\texttt{fermeture}\}, \{\texttt{couverture}, \texttt{non-réplication}\}, <)$. Le résultat suivant établit sous certaines conditions sur la fonction objectif que tout RMDP satisfaisable admet une solution dont les c-blocs sont maximaux pour la relation de réplication. On note $\Pi \lesssim \Pi' \Leftrightarrow (\neg(\Pi < \Pi') \Rightarrow \neg(\Pi' < \Pi))$.

Théorème 1 *Soit* $P = (D, R^+, R^-, <)$ *un RMDP,* C *l'ensemble des c-blocs de* P *et* $C_{\geq_{top}}$ *l'ensemble des c-blocs maximaux pour la réplication. Soit* $f : C \rightarrow C_{\geq_{top}}$ *telle que* $f(c) \geq (c)$ *pour tout* $c \in C$ *et* $F : 2^C \rightarrow 2^{C_{\geq_{top}}}$ *telle que* $F(\{c_1, \ldots, c_k\}) = \{f(c_1), \ldots, f(c_k)\}$. *Si* $F(\Pi) \lesssim \Pi$ *pour tout* $\Pi \in 2^C$ *et* P *est satisfaisable alors* P *a une solution dans* $2^{C_{\geq_{top}}}$.

La condition du théorème est vérifiée si $<$ est le préordre déterminé par la mesure d'inclusion, c'est-à-dire, $\Pi < \Pi' \Leftrightarrow inc_{R^-}(\Pi) < inc_{R^-}(\Pi')$. Elle est aussi vérifiée si la cardinalité minimum est le second critère, c'est-à-dire, $\Pi < \Pi' \Leftrightarrow (inc_{R^-}(\Pi) < inc_{R^-}(\Pi') \vee (inc_{R^-}(\Pi) = inc_{R^-}(\Pi') \wedge card(\Pi) < card(\Pi')))$. Il en est de même si l'on recherche à maximiser la variabilité de motifs. Dans ces cas de figure, le résultat indique qu'on peut se limiter au calcul des c-blocs maximaux par réplication pour obtenir un motif solution. En outre, les c-blocs maximaux par réplication étant nécéssairement fermés sur la classe positive, la contrainte de fermeture peut être ignorée.

D'un point de vue opérationnel, ce résultat motive une approche en deux étapes qui consiste d'abord à calculer les c-blocs maximaux puis à déterminer un motif solution par optimisation combinatoire. La section suivante présente une approche semblable pour le cas général des RMDP où la fonction objectif ne satisfait pas nécessairement aux conditions du théorème.

3 Méthode de résolution RMDP par décomposition

Cette section présente une méthode pour résoudre tout RMDP dont la fonction objectif combine les critères suivants par ordre de priorité : mesure d'exclusion maximum, variabilité minimale et cardinalité minimale. On impose aussi une borne maximum sur la variabilité de motif solution. Une solution est donc un motif fermé sur la classe positive, sans-réplications et minimal pour la fonction objectif. Un motif solution exclut donc le plus grand nombre possible de séquences négatives.

La méthode procède en deux étapes. La première consiste à calculer un ensemble de c-blocs fermés sur la classe positive et qui est sans-inclusions. La seconde identifie pour chacune des séquences négatives lesquels de ces c-blocs ne la couvrent pas, puis détermine un motif solution.

3.1 Calcul de c-blocs

L'algorithme 1 calcule en trois étapes un ensemble sans-inclusions de c-blocs fermés :

1. La première étape (lignes 1–2) est effectuée par l'algorithme 2. Cet algorithme calcule pour la plus petite séquence positive l'ensemble des blocs qui sont de taille 2 (à deux caractères solides), de variabilité inférieure au maximum autorisé $maxslack$, et dont les patrons couvrent la classe positive.
2. La seconde étape (ligne 3) est effectuée par l'algorithme 3. Cet algorithme calcule pour la plus petite séquence positive l'ensemble des blocs dont la variabilité est inférieure au maximum autorisé, dont les patrons couvrent la classe positive, et qui sont maximaux pour l'inclusion dans cet ensemble. Ce calcul est effectué à partir des blocs précédemment calculés.
3. La dernière étape (ligne 4) est effectuée par l'algorithme 4. Cet algorithme calcule à partir des blocs précédemment calculés un ensemble sans-inclusions de c-blocs fermés.

Algorithme 1
`ComputeClosedCBlocks(D+,maxSlack)`

Require: D^+ est la classe positive ; `maxSlack` est la variabilité maximum autorisée.
1: $s \leftarrow \arg\, min_{t \in D^+} |t|$
2: `minbs` $\leftarrow$ `FindAllMinimalBlocks`$(D^+, \texttt{s}, \texttt{maxSlack})$
3: `maxbs` $\leftarrow$ `FindAllMaximalBlocks`$(D^+, \texttt{s}, \texttt{maxSlack}, \texttt{minbs})$
4: `maxcbs` $\leftarrow$ `FindMaximalCBlocks`$(D^+, \texttt{s}, \texttt{maxbs})$
5: **return** `maxcbs`

L'algorithme 2 génère pour la plus petite séquence positive s l'ensemble `minbs` des blocs minimaux dont la couverture est D^+. Pour chaque valeur de variabilité autorisée et chacun des blocs de s dont le patron vérifie $|\pi(p)| = 2$ (ligne 3), l'algorithme vérifie si la couverture de p est D^+ (ligne 4). Lorsqu'un tel bloc est détecté, l'ensemble `minbs` est mis à jour (ligne 5). Soient $n = |D^+|$, $\underline{d}$ et $\overline{d}$ respectivement les longueurs minimum et maximum des séquences positives. La variabilité d'un bloc est bornée par $\underline{d} - 2$ car la séquence s est la plus petite dans D^+. De plus, le nombre maximum de blocs minimaux pour n'importe quelle valeur

Algorithme 2
`FindAllMinimalBlocks`(D^+,s,`maxSlack`)

Require: D^+ est la classe positive, s est une séquence positive, et `maxSlack` est la variabilité maximum autorisée.
1: `minbs` $\leftarrow \varnothing$
2: **for all** $\sigma \in$ [`maxSlack`] **do**
3: **for all** $b = (p, l, s)$ s.t. $|\pi(p)| = 2 \wedge \sigma = slack(p)$ **do**
4: **if** $\varphi(p) = D^+$ **then**
5: `minbs` $\leftarrow$ `minbs` $\cup \{b\}$
6: **return** `minbs`

Algorithme 3
`FindAllMaximalBlocks`(D^+,s,`maxSlack`,`minbs`)

Require: D^+ est la classe positive, s est une séquence positive, `maxSlack` est la variabilité maximum autorisée, et `minbs` est l'ensemble des blocs minimaux de s à couverture positive.
1: `maxbs` $\leftarrow$ `minbs`
2: **repeat**
3: `oldbs` $\leftarrow$ `maxbs`
4: `newbs` $\leftarrow \varnothing$
5: **for all** $\{b' \leftarrow (p', l', s), b'' \leftarrow (p'', l'', s)\} \subseteq$ `oldbs` **do**
6: {For a block $b = (p, l, s)$ we denote $\mu(b) = \{l + k | k \in \pi(p)\}$}
7: $b \leftarrow (p, min(l', l''), s)$ such that $\mu(b) = \mu(b') \cup \mu(b'')$
8: $\sigma \leftarrow slack(p)$
9: **if** $\sigma \leqslant$ `maxSlack` $\wedge\ b \notin$ `newbs` $\wedge\ b \notin$ `oldbs` **then**
10: **if** $\varphi(p) = D^+$ **then**
11: `newbs` $\leftarrow$ `newbs` $\cup \{b\}$
12: **if** `newbs` $\neq \varnothing$ **then**
13: `maxbs` $\leftarrow$ `newbs`
14: **for all** $b \in$ `oldbs` **do**
15: **if** $\nexists b' \in$ `newbs` such that $b \sqsubseteq b'$ **then**
16: `maxbs` $\leftarrow$ `maxbs` $\cup \{b\}$
17: **until** `newbs` $= \varnothing$
18: **return** `maxbs`

de variabilité est borné par $\underline{d} - 1$. En outre, vérifier que la couverture d'un patron est D^+ est de coût inférieur à $n\,\overline{d}$. La complexité en pire cas de `FindAllMinimalBlocks` est donc $\mathcal{O}(n\,\underline{d}^2\,\overline{d})$.

L'algorithme 3 calcule l'ensemble `maxbs` des blocs de la séquence positive s dont les patrons couvrent la classe positive et qui sont maximaux pour l'inclusion dans cet ensemble. L'idée est de construire cet ensemble incrémentalement par fusion de blocs déjà générés. `maxbs` est initialisé à l'ensemble pré-calculé des blocs minimaux (ligne 1). A chaque itération (lignes 2–17), l'ensemble des blocs déjà calculés, noté `oldbs`, est examiné pour construire les blocs du niveau suivant, noté `newbs`. Initialement, `oldbs` est fixé à `maxbs` et `newbs` à l'ensemble vide (lignes 3–4). Un nouveau bloc b est généré (ligne 7) en fusionnant chaque paire de blocs du niveau courant (ligne 5). Si sa variabilité est inférieure à la valeur maximale autorisée et si ce bloc est nouveau (ligne 9), l'algorithme vérifie alors si son patron est positif (ligne 10). Le cas échéant, le bloc est ajouté à l'ensemble `newbs` (ligne 11). Si au moins un bloc nouveau a été généré (ligne 12), l'algorithme élimine tout bloc de `oldbs` qui n'est plus maximal pour l'inclusion (lignes 13–16).

Le nombre d'itérations réalisées par la boucle de la ligne 2 est borné par $\underline{d}$ car le nombre maximum de caractères solides des blocs de `newbs` ne peut que croître à chaque itération et $\underline{d}$ est la taille de la séquence s sur

Algorithme 4 `FindMaximalCBlocks`(D^+,s,`maxbs`)

Require: D^+ est la classe positive, s est une séquence positive, et `maxbs` est l'ensemble des blocs de s à couverture positive et maximaux pour l'inclusion.
1: `maxcbs` $\leftarrow \varnothing$
2: **for all** $(p, l, s) \in$ `maxbs` **do**
3: $\forall t \in D^+\ \ \delta(t) \leftarrow \varnothing$
4: $\delta(s) \leftarrow l$
5: `cbFound` $\leftarrow$ `TRUE`
6: **for all** $t \in D^+$ such that $t \neq s$ **do**
7: **if** $\exists (p, l', t)$ such that
$\forall (p', \delta') \in$ `maxcbs` $(p, l', t) \not\sqsubseteq (p', \delta'(t), t)$ **then**
8: $\delta(t) \leftarrow l'$
9: **else**
10: `cbFound` $\leftarrow$ `FALSE`
11: **if** `cbFound` **then**
12: `maxcbs` $\leftarrow$ `maxcbs` $\cup \{(p, \delta)\}$
13: **return** `maxcbs`

laquelle nous calculons les blocs maximaux. Soit m le nombre maximum de blocs trouvés lors d'une itération. Le nombre d'itérations réalisées par la boucle de la ligne 5 est borné par m^2. Vérifier si un bloc existe déjà est de coût inférieur à m et vérifier si un bloc a une couverture positive est de coût inférieur à $n\overline{d}$. La complexité en pire cas de `FindAllMaximalBlocks` est donc $\mathcal{O}(\underline{d}\,m^2(m + n\,\overline{d}))$.

L'algorithme 4 calcule un ensemble sans-inclusions `maxcbs` de c-blocs fermés sur la classe positive. Le nombre maximum de c-blocs est borné par |`maxbs`| car chacun des blocs $b \in$ `maxbs` de la séquence s peut-être associé à un c-bloc au plus. Pour chaque bloc (p, l, s) précalculé (ligne 2), l'algorithme tente de trouver un c-bloc (lignes 3–12). Il crée d'abord une fonction d'enracinement δ nulle (ligne 3), puis fixe la valeur de $\delta(s)$ à la position l (ligne 4). Il essaie ensuite pour chaque séquence positive t de trouver un bloc dont le patron est p et qui n'est pas inclus dans les blocs de c-blocs précédemment obtenus (lignes 6–7). Si l'algorithme parvient à définir totalement δ alors le nouveau c-bloc est ajouté à l'ensemble `maxcbs`. Soit $m =$ |`maxbs`|, le coût pour trouver un bloc maximal dans chacune des séquences est de $\overline{d}$, et le coût pour vérifier si un bloc a déjà été découvert est de m. La complexité en pire cas de `FindMaximalCBlocks` est donc $\mathcal{O}(mn\,(\overline{d} + m))$.

3.2 Calcul de motif optimal

Une fois calculé l'ensemble sans-inclusions de c-blocs fermés, la seconde étape de la méthode de résolution RMDP détermine un motif optimal. Cette étape se ramène à la résolution d'un problème de couverture minimum par ensembles. Nous en présentons une modélisation sous la forme d'un problème d'optimisation sous contraintes.

Variables et Contraintes. Soit C l'ensemble des c-blocs calculés. Pour chaque séquence négative $s \in D^-$, nous calculons le sous-ensemble des c-blocs de C qui ne couvrent pas s. Cet ensemble est noté $E_s \subseteq C$. Nous associons à chaque c-bloc $i \in C$ une variable $x_i \in \{0, 1\}$

Tableau 1 – Résultats obtenus pour le jeu de séquences protéiques LEAP par résolution RMDP.

cid	#proteins	$\underline{d}$	$\overline{d}$	#nonexp	card	slack	length	time1	time2
1	177	117	507	35	13	14	29	66775	237
2	96	122	338	3	9	27	25	276043	402
3	29	86	186	0	1	0	6	92	124
4	83	81	625	690	1	3	2	6745	8
5	60	83	217	0	3	2	8	311	121
6	202	66	843	258	3	6	6	4079	18
7	53	95	341	0	1	4	4	127	23
8	184	136	411	84	2	1	4	7016	17
9	67	78	144	0	1	2	4	45	15
10	76	88	173	2	7	46	18	49962	674
11	24	159	278	0	5	1	13	358	292
12	15	71	117	0	2	0	6	55	57

indiquant si i appartient au motif ($x_i = 1$) ou non ($x_i = 0$).

Pour chaque séquence négative $s \in D^-$, on souhaite sélectionner au moins un c-bloc qui ne couvre pas s s'il en existe. La mesure d'inclusion du motif est donnée par $inc = \sum_{s \in D^-} (\sum_{i \in E_s} x_i = 0)$. La variabilité du motif est égale à la variabilité maximum de ses c-blocs, c'est-à-dire $slack = max_{i \in C}(slack(i).x_i)$ où $slack(i)$ dénote la variabilité du c-bloc i. La cardinalité du motif est donnée par $card = \sum_{i \in C} x_i$.

Objectif. L'objectif est de minimiser par ordre de priorité l'inclusion, la variabilité et la cardinalité du motif.

$$\texttt{minimize}\ card + \alpha \cdot slack + \beta \cdot inc$$

α et β sont des coefficients dont la valeur est fixée de manière à respecter l'ordre entre les trois critères.

4 Résultats empiriques

La méthode présentée ci-dessus calcule un ensemble de c-blocs pour la variabilité maximum autorisée puis un motif solution. Or il n'est pas toujours nécessaire de calculer tous ces c-blocs si l'on parvient à exclure toutes les séquences négatives à moindre variabilité. L'idée est donc d'utiliser une approche paresseuse qui consiste à incrémenter la valeur de variabilité autorisée de 0 jusqu'à sa valeur maximale ($maxSlack$) en appliquant la méthode à chaque pas. Cette boucle s'arrête dès qu'un motif solution parvient à exclure toutes les séquences négatives. Les résultats présentées ci-dessous font référence à cette approche. A noter que l'implémentation actuelle redémarre la résolution depuis le début à chaque itération. On peut néanmoins envisager de rendre l'algorithme plus incrémental.

Nous présentons ci-dessous les résultats de cette approche sur deux bases de séquences protéiques prépartitionnées : Late Embryogenesis Abundant Proteins (LEAP) et Small Heat Shock Proteins (SHSP). La base LEAP [7] contient 1066 séquences réparties en 12 classes tandis que la base SHSP [8] contient 2244 séquences réparties en 23 classes. Les algorithmes ont été programmés en Java. Les expériences ont été réalisées sous distribution Linux et architecture équipée d'un processeur quatre-coeurs de 2.66 GHz avec 3.8 GB de RAM. Tous les tests ont été menés jusqu'au terme.

Les résultats sont détaillés dans les tableaux 1 et 2. Chacune des classes a été traitée tour à tour comme étant la classe positive, les séquences des autres classes formant alors la classe négative. `cid` correspond à l'identifiant de la classe traitée, `#proteins` correspond au nombre de ses séquences, $\underline{d}$ à leur taille minimum et $\overline{d}$ à leur taille maximum. Pour 6 des 12 classes de LEAP et 3 des 23 classes de SHSP, nous n'avons pas trouvé de motif qui excluait la totalité des séquences négatives. La colonne `#nonexp` indique le nombre de séquences négatives qui n'ont pu être exclues par le motif solution pour chacune des classes. A noter qu'aucune séquence négative n'a pu être exclue pour la classe 11 de SHSP. Les mesures de variabilité, cardinalité et longueur [4] des motifs solutions sont données respectivement dans les colonnes `slack`, `card`, et `length`. Les colonnes `time1` et `time2` donnent le temps de calcul en millisecondes utilisés respectivement pour le calcul de c-blocs et pour le calcul de motif optimal.

Notons que nous avons aussi modélisé le problème RMDP en utilisant Minizinc [10]. Toutefois, les modèles obtenus n'ont pu être appliqués qu'à de petites instances de par leur taille réhdibitoire. Les outils comme Miningzinc (CP for Data mining) ont également été considérés mais ils ne proposaient aucune fonctionnalité pour traiter les données séquentielles.

Un RMPD n'impose aucun séquencement sur les c-blocs d'un motif solution. Cette flexibilité peut se révéler utile comme en témoigne le tableau 3. Ce tableau donne l'enracinement d'un motif solution constitué de 4 c-blocs pour

4. La longueur d'un motif est la somme des nombres de caractères solides des c-blocs qui le composent.

Tableau 2 – Résultats obtenus pour le jeu de séquences protéiques SHSP par résolution RMDP.

cid	#proteins	$\underline{d}$	$\overline{d}$	#nonexp	card	slack	length	time1	time2
1	237	130	163	0	10	10	20	4122	511
2	107	129	174	0	3	2	7	760	258
3	47	165	328	0	5	3	13	1215	388
4	16	119	173	0	3	0	10	251	226
5	14	172	203	0	2	0	6	356	275
6	65	127	248	0	7	8	15	3569	531
7	80	163	266	0	4	3	9	1112	412
8	15	115	146	0	6	1	14	597	357
9	146	121	170	0	4	2	10	236	437
10	294	120	498	1747	1	1	2	5471	9
11	295	130	316	1949	-	-	-	4815	0
12	257	149	269	0	8	25	16	3800	274
13	119	174	271	0	1	1	4	420	332
14	25	145	178	0	1	0	5	473	467
15	25	108	262	0	1	0	7	492	554
16	31	114	154	0	1	0	4	215	257
17	69	102	131	0	3	0	7	96	104
18	154	107	277	77	1	0	2	7765	36
19	23	194	220	0	4	1	9	665	483
20	90	214	298	0	9	3	18	656	251
21	96	161	344	0	3	1	6	217	149
22	13	332	453	0	2	0	7	1474	1392
23	26	79	127	0	2	1	5	342	161

Tableau 3 – Un motif optimal pour la 19^{me} classe de la base SHSP. La cardinalité du motif vaut 4, sa variabilité 1 et sa longueur 9.

protein	PE#V	D#K	Q#S	CS
1	149	90	8	159
2	140	91	8	3
3	140	91	8	3
4	140	91	8	3
5	140	91	8	3
6	147	98	8	157
7	140	91	8	3
8	140	91	8	3
9	147	98	8	157
10	144	75	8	3
11	151	127	69	156
12	137	83	157	142
13	144	75	8	154
14	152	98	8	157
15	141	87	8	146
16	148	89	8	17
17	150	71	8	17
18	140	91	8	3
19	141	122	189	151
20	141	122	189	151
21	140	91	8	3
22	134	85	172	144
23	141	92	81	151

une des classes SHSP. Chaque ligne correspond à l'une des séquences positives et chaque colonne donne la position de départ d'un des c-blocs dans cette séquence. On remarque que les c-blocs ne suivent pas le même ordre selon les séquences.

5 Conclusion

Nous avons présenté le formalisme MDP pour le calcul de motifs discriminants sur données séquentielles prépartitionnées. Ce problème revêt un intérêt particulier en Bio-informatique pour valider a posteriori la classification experte de séquences protéiques en signant chaque classe par un motif exclusif. Le formalisme MDP repose sur un modèle de motifs à base de c-blocs qui permet de distinguer contraintes sur c-blocs et contraintes entre c-blocs. Il permet en outre de limiter le nombre de contraintes à traiter selon leurs propriétés (anti-monotonie, subsomption, etc). Dans ce cadre, nous avons présenté plusieurs contraintes visant à éliminer des motifs redondants (fermeture, non-inclusion) ou à satisfaire des besoins applicatifs (non-réplication, séquentialité).

Nous nous sommes intéressés à la classe RMDP où l'on recherche des motifs fermés sur la classe positive, sans-réplications et couvrant minimalement la classe négative. Sous certaines conditions portant sur la fonction objectif, nous avons établi qu'il suffisait de se limiter au calcul de c-blocs maximaux par réplication afin d'obtenir un motif solution. Dans le cas général, nous avons proposé une mé-

thode en deux étapes consistant à construire des c-blocs fermés puis à résoudre un problème de couverture par ensembles. L'efficacité de cette méthode a été démontrée sur des bases de séquences protéiques.

Nous entendons poursuivre ce travail dans diverses directions : mise en oeuvre de nouvelles contraintes sur motifs, étude de la complexité théorique de différentes classes MDP selon le langage de contraintes, correspondance avec le problème d'alignement de séquences et comparaison expérimentale avec des algorithmes dédiés, et extension au problème de partitionnement automatique (clustering).

Références

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

[2] Emmanuel Coquery, Said Jabbour, and Lakhdar Saïs. A constraint programming approach for enumerating motifs in a sequence. In *2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW)*, pages 1091–1097, 2011.

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

[4] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(2) :402–418, 2013.

[5] Dan Gusfield. *Algorithms on strings, trees, and sequences : computer science and computational biology*. Cambridge University Press, Cambridge [England] ; New York, 2008.

[6] Gilles Hunault and Emmanuel Jaspard. LEAPdb : a database for the late embryogenesis abundant proteins. *BMC Genomics*, 11(1) :221, April 2010. PMID : 20359361.

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

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

[9] Jean-Philippe Métivier, Samir Loudni, and Thierry Charnois. A constraint programming approach for mining sequential patterns in a sequence database.

[10] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc : Towards a standard cp modelling language. In *Principles and Practice of Constraint Programming–CP 2007*, pages 529–543. Springer, 2007.