Découverte de motifs graduels partiellement ordonnés : application aux données d'expériences scientifiques

Simon Ser*, Fatiha Saïs*, Maguelonne Teisseire**

*LRI, Université Paris Sud, Bât. 650-Ada Lovelace,
91405 Orsay Cedex, France
simon.ser@emersion.fr, Fatiha.Sais@lri.fr
http://www.lri.fr/~sais
** TETIS Irstea Université de Montpellier,
500, rue J. F. Breton 34093, Montpellier Cedex 5, France
maguelonne.teisseire@irstea.fr
http://textmining.biz/Staff/Teisseire/

Résumé. Les données séquentielles sont aujourd'hui omniprésentes et concernent divers domaines d'application. La fouille de données de séquences permet d'extraire des informations et des connaissances pouvant être à forte valeur ajoutée. Cependant, lorsque les données de séquences sont riches en données numériques, des méthodes de fouille de données plus fines sont nécessaires pour extraire des connaissances plus expressives représentant la variabilité des valeurs numériques ainsi que leur éventuelle interdépendance. Dans cet article, nous présentons une nouvelle méthode de découverte de séquences graduelles fréquentes représentées par des graphes à partir d'une source de données de séquences en RDF (Resource Description Framework [1]). Ces dernières sont transformées en graphes graduels partiellement ordonnés, *gpo*. Nous proposons un algorithme permettant de découvrir les sous-graphes *gpo* fréquents. Une expérimentation sur deux jeux de données réelles ont montré la faisabilité et la pertinence de notre approche.

1 Introduction

En raison du développement du marché des objets connectés et des techniques de géolocalisation, les données séquentielles sont omniprésentes et produites en quantité de plus en plus importante. Elles concernent une multitude de domaines d'application, allant de la médecine jusqu'aux télécommunications en passant par l'éducation (Kumar et al. (2011)). Les données séquentielles peuvent être produites sous la forme de suites d'informations ordonnées (e.g., parcours de patients, séquences de génomes, expériences scientifiques) ou sous la forme de séries temporelles (e.g., analyse du signal, données météorologiques, économétrie).

La fouille de motifs séquentiels, qui consiste à découvrir des sous-séquences fréquentes à partir d'une base de données séquentielles, a suscité un grand intérêt dans une multitude de

1. https://www.w3.org/RDF/

domaines. Par exemple, en télécommunication, des motifs séquentiels sur des mouvements de groupes d'utilisateurs d'appareils mobiles peuvent être utilisés pour prédire la position future d'un utilisateur. Dans le domaine bioinformatique on peut rechercher des motifs dans des séquences d'ADN ou des motifs séquentiels permettant la prédiction de la fonction de certaines protéines. Dans le domaine des sciences expérimentales, les scientifiques recherchent dans les comptes-rendus d'expériences des sous-séquences d'étapes fréquentes permettant la prédiction de valeurs d'observations manquantes ou encore des relations de cause-à-effet. Dans cet article, nous avons choisi ce dernier domaine d'application pour définir, illustrer et évaluer l'efficacité de notre approche de découverte de séquences graduelles partiellement ordonnées.

Les expériences scientifiques réalisées sont souvent répétées plusieurs fois dans des conditions différentes. Le volume et les proportions des produits utilisés, ainsi que la configuration du milieu de l'expérimentation varient. Ainsi, trouver des observations similaires entre plusieurs expériences n'est pas envisageable. Par contre, dans ce contexte il est pertinent de rechercher des évolutions similaires entre deux observations. Ces augmentations ou diminutions des valeurs des observations sont appelées gradualité. Aussi, une expérience ne se déroule pas de manière linéaire : certaines manipulations sont réalisées en parallèle avec d'autres ce qui constitue une information importante sur l'ordre de déroulement des étapes de l'expérience qui doit être prise en compte. En effet, il faut conserver la coexistence des observations, i.e., il n'existe pas d'ordre total sur celles-ci, mais uniquement un ordre partiel. Comme étape préalable à la découverte de règles causales nous proposons une méthode automatique qui extrait des motifs séquentiels fréquents. Ces motifs sont graduels, puisque qu'ils expriment la diminution et l'augmentation des valeurs des attributs ; ils sont partiellement ordonnés car ils représentent l'ordre partiel pouvant être présent dans les données initiales. L'approche que nous présentons dans cet article prend en entrée une base de séquences valuées partiellement ordonnées sous la forme d'une base de graphes orientés étiquetés aux sommets et fournit en sortie un ensemble de motifs séquentiels *graduels* partiellement ordonnés représentés par des graphes orientés étiquetés aux arcs. La méthode développée a été testée et évaluée sur deux jeux de données réelles en biologie. Le premier concerne des données issues du projet Qualiment CellExtraDry sur les processus de transformation et de stabilisation de la levure ; et le second concerne des données issues du projet Carrédas sur des expériences sur la dégustation et la digestion de gels laitiers.

Cet article est organisé de la façon suivante. Nous présentons tout d'abord un rapide panorama des méthodes qui s'intéressent à la découverte de séquences fréquences et de motifs graduels. Nous posons ensuite quelques définitions avant de détailler notre proposition ainsi que les algorithmes associés. Nous présentons et discutons les expérimentations réalisées sur des données réelles. Nous concluons par un bilan et des perspectives à court et moyen terme.

2 État de l'art

Les recherches sur l'extraction de connaissances à partir de données séquentielles restent peu nombreuses au sein de la communauté fouille de données même si un regain se fait ressentir du fait de la mise à disposition de nombreuses séries temporelles (comme par exemple dans le domaine de l'imagerie satellitaire). Nous pouvons citer Pei et al. (2004) qui traite le problème de l'extraction de motifs séquentiels à partir d'une base de données de séquences à l'aide de l'algorithme *PrefixSpan*, en utilisant une technique de croissance de motif. Afin

de lever la contrainte forte de précédence, Pei et al. (2006) propose l'algorithme *Frecpo* pour extraire des motifs partiellement ordonnés de séquences simples sans items répétés et sans itemsets. Malgré les bonnes performances de *Frecpo*, son champ d'application est très limité étant donné que la même information peut apparaître plusieurs fois ou plusieurs informations peuvent être datées en même temps dans une base de données.

Dans Casas-Garriga (2005), un algorithme est présenté pour extraire des motifs partiellement ordonnés (po) clos (c'est-à-dire que pour chaque motif extrait, il n'existe pas de sous-séquence ayant le même support) d'une base de séquences. Il extrait d'abord des motifs séquentiels, puis transforme les motifs séquentiels en motifs po en post-traitement. Il n'extrait donc pas directement les motifs po. D'autre part, d'après Fabrègue et al. (2015), il n'extrait qu'un sous-ensemble des motifs po.

Dans Fabrègue et al. (2015), l'algorithme *OrderSpan* est décrit et permet d'extraire des motifs po clos d'une base de séquences directement, en utilisant comme dans Pei et al. (2004) une technique de croissance de motif. *OrderSpan* est efficace et peut passer à l'échelle, néanmoins il prend en entrée une base de données de séquences non po et n'est pas capable de gérer les items graduels.

Berzal et al. (2007) adapte *Apriori* (Agrawal et Srikant, 1994) afin d'extraire des motifs graduels d'une base de données de valeurs. D'une part, cette méthode ne passe pas à l'échelle en raison de sa complexité algorithmique et d'autre part, elle ne s'intéresse pas aux informations datées. Dans Di-Jorio et al. (2009), un algorithme nommé *GRITE* permet d'extraire plus efficacement les motifs graduels, mais ne s'intéresse également pas aux contraintes de temporalité.

Lonlac et al. (2017) présente un processus d'extraction de motifs graduels fermés sous contrainte de temporalité. Les auteurs utilisent *Apriori* (Agrawal et Srikant, 1994) mais un autre algorithme plus performant comme *PrefixSpan* (Pei et al., 2004) pourrait le remplacer. Néanmoins, la contrainte de temporalité est appliquée en post-traitement en filtrant les motifs produits ce qui alourdit le processus. Ajoutons aussi que la base de données d'entrée est très simple et peut être vue comme une seule séquence valuée non partiellement ordonnée.

Nous pouvons constater qu'il n'existe pas d'approche permettant d'extraire directement des motifs graduels po d'une base de données de séquences. C'est ce que nous proposons dans cet article en adoptant l'algorithme Fabrègue et al. (2015) en utilisant également une technique de croissance de motif.

3 Motifs graduels partiellement ordonnés

Nous allons tout d'abord présenter quelques définitions en les illustrant à l'aide de notre domaine d'application : les expériences scientifiques. Ensuite, nous décrirons notre algorithme.

3.1 Concepts et définitions

Une expérience est composée de plusieurs étapes, chaque étape ayant des observations. Une observation possède une qualité et une valeur (par exemple la qualité *température* et la valeur *42 degrés*) et peut être vue comme un item valué. Une étape peut être modélisée par un itemset valué.

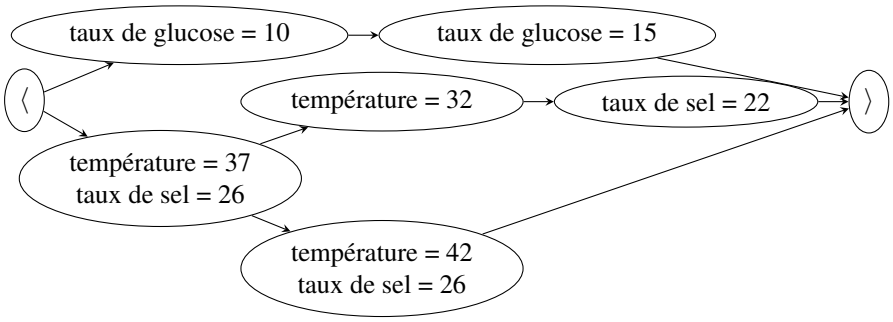


FIG. 1: Un exemple d'expérience

Définition 1 (Item valué) *Soit $\mathcal{I}$ un ensemble d'items. Un **item valué** I_v est un couple (I, v) avec $I \in \mathcal{I}$ et $v \in \mathbb{R}$. Un **itemset valué** IS_v est un ensemble non vide et non ordonné d'items valués vérifiant :*

$$\forall (I, v), (I', v') \in IS_v, I \neq I'$$

On note $\mathcal{I}_v = \mathcal{I} \times \mathbb{R}$ l'ensemble des items valués et $\mathcal{IS}_v = \mathcal{P}(\mathcal{I}_v) \setminus \{\varnothing\}$ l'ensemble des itemsets valués.

Une évolution entre deux items valués (augmentation, diminution, stagnation) peut être représentée par un item graduel. On dit alors que les deux items valués supportent l'item graduel.

Définition 2 (Item graduel) *Un **item graduel** I_g est un item $I \in \mathcal{I}$ muni d'un symbole $\circ \in \{<, >, =\}$, noté I°. On note $\mathcal{I}_g$ l'ensemble des items graduels :*

$$\mathcal{I}_g = \left\{ I^\circ; I \in \mathcal{I}, \circ \in \{<, >, =\} \right\}$$

Définition 3 (Item graduel supporté par un couple d'items valués) *Soit un couple d'items valués (I_v, I'_v) tel que $I_v = (I, v)$ et $I'_v = (I, v')$. Ce couple **supporte** l'item graduel I° si et seulement si $v \circ v'$.*

Les étapes d'une expérience ne se déroulent pas nécessairement l'une après l'autre, par exemple deux étapes peuvent se dérouler en parallèle. On peut donc représenter une expérience par un graphe dont les sommets sont les étapes. Un exemple d'un tel graphe est illustré figure 1. Ce graphe est appelé une séquence valuée partiellement ordonnée.

Définition 4 (Séquence valuée partiellement ordonnée) *Une **séquence valuée partiellement ordonnée** (séquence valuée po, ou vpo) est un graphe orienté acyclique étiqueté aux sommets $G = (\mathcal{V}, \mathcal{A}, \Sigma_\mathcal{V}, l_\mathcal{V})$ où :*

- *$\mathcal{V}$ est l'ensemble des sommets et $\mathcal{A}$ est l'ensemble des arcs où $(\mathcal{V}, \prec)$ est un ordre partiel sur les sommets et $\mathcal{A} = \{(u, v) \in \mathcal{V}^2 \mid (u, v) \in \prec\}$*
- *$\Sigma_\mathcal{V} = \mathcal{IS}_v$ est l'alphabet des étiquettes des sommets, soit l'ensemble des itemsets valués*
- *$l_\mathcal{V} : \mathcal{V} \to \Sigma_\mathcal{V}$ donne la correspondance entre les sommets et leur étiquette*

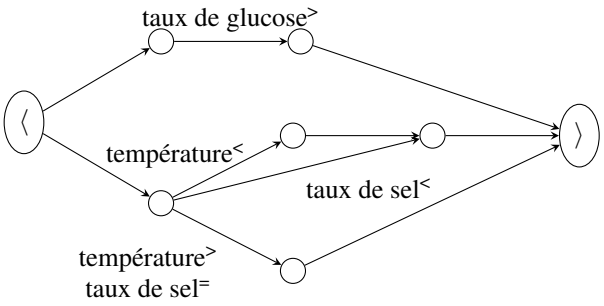


FIG. 2: La séquence graduelle po correspondant à la séquence valuée po de la figure 1

Les deux sommets étiquetés par $\langle$ et $\rangle$ sont utilisés pour marquer le début et la fin de la séquence po.

Pour u, v deux sommets du graphe, $u \prec v$ si et seulement s'il y a un chemin de u à v. S'il n'y a pas de chemin de u à v, alors ces deux éléments ne sont pas comparables.

Si une étape se déroule avant une autre, on peut lister les items graduels représentant les évolutions des observations entre ces deux étapes. On dit alors que la séquence supporte les items graduels.

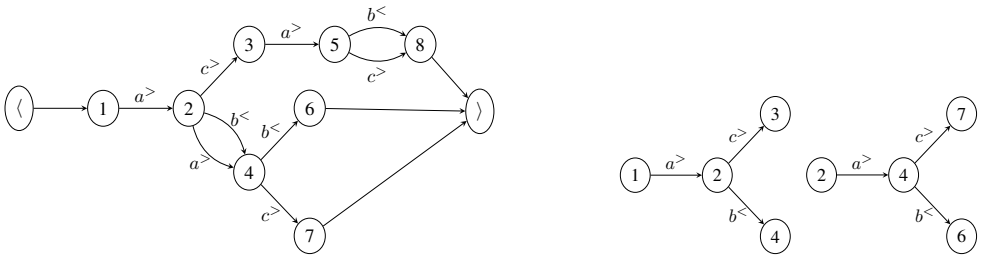
Définition 5 (Item graduel supporté par une séquence valuée po) *Une séquence valuée po $G_v = (\mathcal{V}, \mathcal{A}, \Sigma_\mathcal{V}, l_\mathcal{V})$ supporte un item graduel I_g s'il existe deux sommets $V \prec V' \in \mathcal{V}$ tel qu'il existe deux items valués $I_v \in l_\mathcal{V}(V)$ et $I'_v \in l_\mathcal{V}(V')$ tel que le couple d'items valués (I_v, I'_v) supporte I_g.*

Dans le graphe d'une expérience, les évolutions des observations entre les étapes sont représentées par des arcs étiquetés par l'item graduel associé à cette évolution. Par exemple, d'un sommet *la température vaut 36 degrés* à un sommet *la température vaut 42 degrés*, on peut définir un arc *la température augmente*. Le graphe obtenu est appelé séquence graduelle partiellement ordonnée. La figure 2 illustre la séquence graduelle po correspondant à la séquence valuée po de la figure 1.

Définition 6 (Séquence graduelle partiellement ordonnée) *Une **séquence graduelle partiellement ordonnée** (séquence gpo) est un graphe orienté acyclique étiqueté aux arcs $G = (\mathcal{V}, \mathcal{A}, \Sigma_\mathcal{A}, l_\mathcal{A})$ où :*

- *$\mathcal{V}$ est l'ensemble des sommets et $\mathcal{A}$ est l'ensemble des arcs*
- *$\Sigma_\mathcal{A} = \mathcal{I}_g \cup \{\varnothing\}$ est l'alphabet des étiquettes des arcs, soit l'ensemble des items graduels (ou $\varnothing$ pour signifier une absence d'étiquette)*
- *$l_\mathcal{A} : \mathcal{A} \to \Sigma_\mathcal{A}$ donne la correspondance entre les arcs et leur étiquette*

Pour signifier une absence d'évolution de la valeur d'un item (donc une absence d'item graduel) mais pour conserver l'ordre de deux sommets, on veut pouvoir dessiner un arc entre ces sommets sans l'étiqueter. L'étiquette $\varnothing$ est alors utilisée.



(a) Une séquence graduelle po S (b) Exemples de mineurs de S

FIG. 3: Exemples de mineurs d'une séquence graduelle po

3.2 Support

Notre but est d'extraire un ensemble d'évolutions communes à plusieurs expériences. Plus précisément, nous souhaitons extraire les évolutions qui sont suffisamment fréquentes, que nous appellerons *motifs graduels po*. Pour cela, nous allons définir la notion de support.

Il s'agit de compter le nombre de séquences dans la base de données qui supportent un motif donné. La notion de M-inclusion est adoptée pour définir formellement l'occurrence du motif dans une séquence. Deux graphes sont m-inclus s'il est possible de passer de l'un à l'autre en supprimant des arêtes ou sommets et en fusionnant des sommets.

Définition 7 (M-inclusion) *Soit S, S_p des séquences po. On dit que S_p est **m-inclus** dans S si et seulement s'il existe un mineur (Lovász, 2005) de S isomorphe à S_p. On note $S_p \preceq_m S$.*

La figure 3 illustre cette définition.

Définition 8 (Support en nombre de séquences) *Soit DB une base de données de séquences po et S_p une séquence po. Le **support** de la séquence S_p par la base DB est défini par :*

$$support(DB, S_p) = Card\{S \in DB \mid S_p \preceq_m S\}$$

3.3 Algorithmes

Étant donnée une base de données de séquences valuées po, *GradualSpan* va extraire l'ensemble des séquences graduelles po ayant un support supérieur à un support minimum θ donné. Ces séquences extraites sont appelées des motifs graduels po. Notre algorithme fonctionne en trois étapes (la dernière n'est pas développée dans cet article) :

— **Transformation des items valués en items graduels** : chaque séquence valuée po de la base de données est convertie en séquence graduelle po.

— **Découverte des motifs graduels po** : l'algorithme (Fabrègue et al., 2015) *OrderSpan* est modifié pour accepter une base de données de séquences graduelles po en entrée à la place de séquences po. Il produit des motifs graduels po en sortie.

3.3.1 Transformation des items valués en items graduels

La conversion d'une séquence valuée po en une séquence graduelle po est réalisée à l'aide de l'algorithme 1 par un parcours en profondeur (DFS).

Algorithme 1 VALUEDTOGRADUAL

Entrée : S_v une séquence valuée po de racine n_v, $last$ un dictionnaire des derniers items valués rencontrés et $visited$ un dictionnaire des sommets visités
Sortie : S_g la séquence graduelle po de racine n_g associée à S_v

1 **if** $visited$ contains the key n_v **then**
2 $n_g \leftarrow visited$.GET(n_v)
3 **else**
4 $n_g \leftarrow$ NEWNODE()
5 $visited$.PUT(n_v, n_g)
6 **for all** I_v valued item of $n_v.itemset$ **do**
7 **if** $last$ contains the key $I_v.item$ **then**
8 $(v_{last}, n_{last}) \leftarrow last$.GET($I_v.item$)
9 $I_g \leftarrow$ NEWGRADUALITEM($I_v.item$, v_{last}, $I_v.value$)
10 create an arc from n_{last} to n_g with label I_g
11 $last \leftarrow last$.PUT($I_v.item$, $(I_v.value, n_g)$)
12 **for all** n_v^{next} successor of n_v **do**
13 $n_g^{next} \leftarrow$ VALUEDTOGRADUAL(n_v^{next}, $last$, $visited$)
14 create an unlabelled arc from n_g to n_g^{next}
15 **return** n_g

L'algorithme est appliqué récursivement à chaque sommet de la séquence valuée po d'entrée. Tout d'abord, comme il y a autant de sommets dans la séquence vpo que dans la séquence gpo associée, un nouveau sommet est créé (lignes 1-5). Le dictionnaire $visited$ permet de ne créer qu'un sommet pour la séquence graduelle par sommet de la séquence valuée.

Ensuite, on va tracer les arcs entre les sommets existants et le nouveau sommet (lignes 6-11). Pour chaque item valué I_v, on vérifie si on a déjà rencontré son item $I_v.item$. Si oui, on crée l'item graduel I_g à partir de l'item valué déjà rencontré et de l'item valué actuel, puis on dessine l'arc correspondant. Dans tous les cas, on retient dans le dictionnaire $last$ que l'on vient de rencontrer l'item.

Enfin, on rappelle l'algorithme récursivement sur tous les successeurs du sommet actuel (ligne 12). Afin de conserver les arcs de la séquence vpo d'entrée, on dessine des arcs non étiquetés du sommet actuel aux sommets nouvellement créés.

Pendant la traversée du graphe, le dictionnaire $last$ contient tous les items déjà rencontrés lors du parcours du chemin de la racine jusqu'au sommet actuel. Lorsqu'on parcourt les successeurs, chaque appel récursif va déclencher de nouvelles rencontres d'items, or on ne souhaite pas qu'une rencontre d'item dans une branche soit visible dans une autre branche. Pour cela, l'opération d'ajout PUT dans le dictionnaire $last$ ne doit pas le modifier *in-place*. Pour représenter ce dictionnaire, on préférera donc utiliser un arbre de recherche (Knuth (1998)) plutôt qu'une table de hachage (cela permet de réaliser les opérations en temps logarithmique tout en partageant la mémoire).

L'algorithme VALUEDTOGRADUAL consiste en un parcours en profondeur de chaque séquence valuée de la base d'entrée. Néanmoins, afin d'extraire tous les items graduels, il est nécessaire de visiter plusieurs fois les mêmes noeuds, plus précisément de visiter tous les

chemins du graphe orienté acyclique. La complexité dans le pire des cas de VALUEDTOGRADUAL est $\mathcal{O}(|DB_v|.v_{max}.2^{v_{max}})$, avec $|DB_v|$ le nombre de séquences dans la base et v_{max} le nombre de sommets de la séquence la plus grande. Cette complexité est linéaire en nombre de séquences à traiter et exponentielle en la taille des séquences.

3.3.2 Découverte de motifs graduels

La découverte de motifs graduels est réalisée avec l'algorithme 2, une variante de ORDERSPAN (Fabrègue et al., 2015).

Algorithme 2 FORWARDTREEMINING

Entrée : DB_g une base de données de séquences graduelles po, θ un support minimum et $mined$ l'ensemble des sous-ensembles de la base de données déjà explorés
Sortie : $pattern$ un ensemble de motifs graduels po

```
1  pattern ← NEWDAG()
2  pattern.begin.database ← NEWPROJECTEDDATABASE(DB_g)
3  nodeQueue ← NEWQUEUE({pattern.begin})
4  while nodeQueue is not empty do
5      n ← nodeQueue.POP()
6      occList ← LISTOCCURENCES(n.database, θ)
7      for all occ in occList such that occ.support = |DB_g| do
8          n' ← NEWNODE()
9          n'.database ← n.database.PROJECTON(occ.item)
10         draw an arrow from n to n' labelled by occ.item
11         nodeQueue.PUSH(n')
12     for all occ in occList such that occ.support < |DB_g| do
13         DB'_g ← {S ∈ DB_g | S supports occ.item}
14         if DB'_g ∉ mined then
15             mined ← mined ∪ {DB'_g}
16             FORWARDTREEMINING(DB'_g, θ, mined)
17 MERGINGSUFFIXTREE(pattern.end)
18 OUTPUT(pattern)
```

Chaque appel à FORWARDTREEMINING extrait un motif ayant un support égal au support maximal de la base de données. Tout d'abord, un graphe acyclique orienté est créé pour contenir le motif qui sera extrait (ligne 1). On va attacher à chaque sommet une base de données projetée contenant les suffixes des séquences de DB_g par rapport au préfixe (i.e. items rencontrés sur le chemin de la racine jusqu'au sommet actuel). Une base projetée est attachée après tous les items sur lesquels elle a été projetée. Pour le premier sommet, la base de données projetée est exactement la base de données d'entrée (ligne 2). Une file $nodeQueue$ est initialisée avec le premier sommet (ligne 3).

La boucle ligne 4 étend le motif au maximum, i.e. la file est vide lorsque le motif ne peut plus être étendu. La liste des items ayant un support au moins égal à θ est construite par un parcours en profondeur de chaque séquence de la base projetée (ligne 6).

Dans un premier temps, on ne considère que les items qui ont un support égal au cardinal de la base d'entrée (lignes 7-11). Comme on est en train de construire un motif ayant un support égal au support maximal, seuls ces derniers peuvent être ajoutés au motif sans faire diminuer son support. Pour chacun de ces items, on crée un nouveau sommet n' en lui attachant la base de données projetée sur l'item.

Ensuite, on considère les items qui ont un support strictement inférieur au cardinal de la base (lignes 12-16). Ces items vont pouvoir être étendus avec des items du même support pour produire des motifs. Pour cela, on extrait le sous-ensemble DB'_g de la base de données qui supporte l'item, et on se rappelle récursivement sur ce sous-ensemble de la base d'entrée. Pour éviter de parcourir plusieurs fois les mêmes sous-ensembles de la base d'entrée, on utilise l'ensemble $mined$.

Une fois que le motif ne peut plus être étendu, on supprime les redondances du motif en appliquant MERGINGSUFFIXTREE (ligne 17, cet algorithme n'est pas présenté dans ce papier).

Notons que cet algorithme a la propriété d'extraire des motifs graduels po *clos*, c'est-à-dire que pour chaque motif extrait, il n'existe pas de sous-séquence ayant le même support.

La procédure FORWARDTREEMINING a la même complexité qu'ORDERSPAN (Fabrègue et al., 2015) étant donné que le parcours des séquences est strictement identique.

4 Expérimentations

Nous présentons dans cette section les premiers résultats de l'évaluation de notre approche que nous avons effectuée sur des données scientifiques.

Description du jeu de données. Nous avons évalué notre approche de découverte de motifs graduels partiellement ordonnés sur des données scientifiques représentées en RDF et relevant du domaine de l'agro-alimentaire. Ces données sont décrites suivant les classes et les propriétés de l'ontologie PO^2 (Ibanescu et al. (2016)) représentant les processus de transformations [2]. Plus particulièrement, PO^2 contient les classes suivantes :

- *Itinéraire*, une réalisation d'un processus, composée d'un graphe dirigé d'étapes,
- *Mixture*, représente l'objet étudié dans le processus et elle est composée d'un ensemble de produits,
- *Étape*, possède des manipulations et observations sur une mixture à un instant donné,
- *Attribut*, possède une qualité (e.g. volume, poids, température...), une unité et une valeur numérique,
- *Observation*, représente une mesure effectuée à une date donnée, en utilisant une méthode de mesure donnée, concerne une étape, un produit ou une mixture donnée et obtient une ou plusieurs valeurs résultat.

Chaque itinéraire peut être vu comme une séquence valuée po, les attributs étant des items valués.

Dans le cadre du projet INRA CellExtraDry, PO^2 a été enrichie par une partie d'Agrovoc [3]. Elle a été ensuite peuplée par 3 processus de transformation de micro-organismes (levures) suivant 20 itinéraires, 53 étapes (séchage, chauffage ...) et 286 observations. Chaque processus possède 3 à 6 itinéraires, chaque itinéraire possède à son tour une quinzaine d'étapes et environ 70 observations. Les données réelles ne pouvant être mises à disposition, pour des raisons de confidentialité, un échantillon de ces données, décrivant un seul processus de stabilisation et dont certaines valeurs ont été modifiées, est accessible avec le code en $Java$ de l'algorithme et ses résultats sur : `https://github.com/emersion/gradualspan`.

Résultats de l'algorithme GradualSpan. En exécutant *GradualSpan* sur la base de données avec un support minimal égal à 15, on obtient 5 motifs graduels po. Le premier a un support

2. PO^2 est disponible à l'adresse suivante : `http://agroportal.lirmm.fr/ontologies/PO2`
3. Agrovoc est accessible via le lien : `http://aims.fao.org/fr/agrovoc`

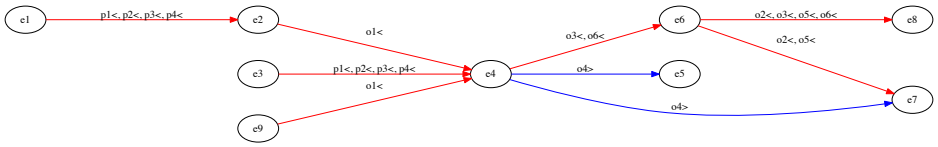
de 20 (il est supporté par toute la base). Les suivants ont un support de 16 et sont tous des variations du premier (ils contiennent des items graduels en plus ce qui fait diminuer le support du motif).

TAB. 1: Les attributs produits, les étapes et les attributs observés impliqués dans les motifs

Attributs-Produits	***Attributs-Observés***	***Étapes***
p_1 : Masse-Glucose	o_1 : consommation-eau	e_1 : préparation milieu pré-culture 2
p_2 : Masse-Eau	o_2 : logarithme-concentration	e_2 : fermentation pré-culture 2
p_3 : Masse-Extrait de Levure	o_3 : concentration	e_3 : préparation milieu G+
p_4 : Masse-kh2po4	o_4 : impulsion	e_4 : culture
	o_5 : logarithme-concentration-par-batch	e_5 : nettoyage du fermenteur
	o_6 : concentration-par-batch	e_6 : séchage à l'étuve
	o_7 : c-33924	e_7 : séchage sur lit fluidisé 45°90°
	o_8 : différence-d-impulsion	e_8 : séchage sur lit fluidisé 60° 60°
	o_9 : conversion-d-impulsion	e_9 : stérilisation du milieu
	o_{10} : énergie-active	

Dans le tableau 1 nous présentons la liste des attributs représentant la composition des mixtures, la liste des attributs observés ainsi que la liste des étapes impliqués dans des variations exprimées dans les cinq motifs découverts dans la source de données.

m_1



m_3

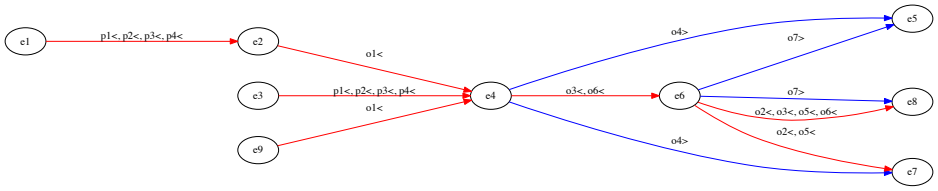


FIG. 4: Deux exemples de motifs extraits : m_1 et m_3

Les motifs produits par GradualSpan sont représentés par un graphe graduel dont les arcs expriment les variations des valeurs des attributs des produits et des attributs observés ; et les nœuds représentent une étape ou une succession d'étapes. Afin de reconstituer les étapes concernées par les variations, nous avons mis en correspondance les motifs obtenus avec les graphes graduels représentant les 20 itinéraires considérés. En figure 4, nous montrons deux exemples de motifs graduels extraits et traduits en considérant les étapes des graphes graduels de départ (les arcs rouges expriment une diminution et les arcs bleus une augmentation). Nous pouvons remarquer que m_1 et un sous-graphe de m_3 et que dans m_3 nous avons deux infor-

mations additionnelles qui expriment le fait qu'on a observé une augmentation de la valeur de l'observation o_7 lorsque l'étape e_6 et suivie de l'étape e_8 ainsi que lorsque e_6 est suivie de e_8.

L'un des objectifs visés par les experts biologistes ayant produit cette source de données est de trouver des relations de causes-à-effets entre les attributs composants les mixtures et les attributs observés. En effet, on pourrait traduire les différents chemins exprimés dans les motifs graduels partiellement ordonnés en règles logiques, dont les prémisses sont variations sur les valeurs des attributs-produits et des observations intermédiaires et la conclusion représenterait les variations détectées pour les observations concernant les dernières étapes dans le graphe. A partir du motifs m_1 on pourrait générer 12 règles comme par exemple, celles qui expriment : R1 : Si $(p_1^< p_2^< p_3^< p_4^<)$ entre les étapes e_3 et e_4 alors $(o_4^>)$ entre les étapes e_4 et e_7.

R2 : Si $(p_1^< p_2^< p_3^< p_4^<)$ entre les étapes e_3 et e_4, et $(o_3^< o_6^<)$ entre les étapes e_4 et e_6 alors $(o_2^< o_3^< o_6^< o_5^<)$ entre les étapes e_6 et e_8.

En résumé, les motifs graduels obtenus sont très expressifs et permettent de représenter des relations de dépendances entre des variations subies par les attributs des produits composant les mixture et ceux des observations. Il est possible de générer des règles logiques exprimant ces dépendances. Dans la suite nous souhaitons, tout d'abord, faire évaluer les motifs obtenus par les experts du domaine afin de rendre compte de leur qualité et pertinence pour le domaine d'application. Nous souhaitons également générer les règles logiques à partir des motifs de façon complètement automatique.

5 Conclusion

La méthode que nous avons développée est inspirée de l'approche OrderSpan Fabrègue et al. (2015) qui prend en entrée une base de séquences et qui produit des séquences partiellement ordonnées. Nous avons proposé un premier algorithme pour transformer les séquences partiellement ordonnées en des séquences graduelles partiellement ordonnées exprimant les variations (augmentation et diminution) des valeurs des attributs. Puis un second algorithme a été développé pour rechercher les sous-séquences graduelles partiellement ordonnées fréquentes, i.e. respectant une valeur de support minimum. Enfin, un post-traitement est ensuite appliqué afin d'éliminer les arcs vides (non porteurs d'informations) mais aussi des arcs redondants ou pouvant être retrouvés par transitivité. Les perspectives associées sont nombreuses. Nous citerons tout d'abord la nécessité d'offrir aux experts une interface aisée de navigation entre les données sources et les motifs po afin de valider ceux-ci. Enfin la prise en compte des itemsets graduels doit être également proposée.

Références

Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases*, 487–499.

Berzal, F., J. Cubero, D. Sanchez, M. Vila, et J. Serrano (2007). An alternative approach to discover gradual dependencies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15*, 559–570.

Casas-Garriga, G. (2005). Summarizing sequential data with closed partial orders. *SIAM International Conference on Data Mining*.

Di-Jorio, L., A. Laurent, et M. Teisseire (2009). Mining frequent gradual itemsets from large databases. *Advances in Intelligent Data Analysis VIII, 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31 - September 2, 2009. Proceedings*, 297–308.

Fabrègue, M., A. Braud, S. Bringay, F. L. Ber, et M. Teisseire (2015). Mining closed partially ordered patterns, a new optimized algorithm. *Knowledge-Based Systems 79*, 68–79.

Ibanescu, L., J. Dibie, S. Dervaux, E. Guichard, et J. Raad (2016). Po^2 - A process and observation ontology in food science. application to dairy gels. In *Metadata and Semantics Research - 10th International Conference, MTSR 2016, Göttingen, Germany, November 22-25, 2016, Proceedings*, pp. 155–165.

Knuth, D. (1998). *The Art of Computer Programming*, Volume 3, Chapter 6.2.2. Binary Tree Searching, pp. 572–611. Addison-Wesley.

Kumar, P., P. Kumar, P. R. Krishna, et S. B. Raju (2011). *Pattern Discovery Using Sequence Data Mining: Applications and Studies* (1st ed.). Hershey, PA, USA: IGI Global.

Lonlac, J., Y. Miras, A. Beauger, M. Pailloux, J.-L. Peiry, et E. M. Nguifo (2017). Une approche d'extraction de motifs graduels (fermés) fréquents sous contrainte de la temporalité. *EGC 2017, vol. RNTI-E-33*, 213–224.

Lovász, L. (2005). Graph minor theory. *Bull. Amer. Math. Soc. (New Series) 43*, 75–86.

Pei, J., J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, et M.-C. Hsu (2004). Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering 16*, 1424–1440.

Pei, J., H. Wang, J. Liu, K. Wang, et P. Yu (2006). Discovering frequent closed partial orders from strings. *IEEE Transactions on Knowledge and Data Engineering 16*.

Summary

The sequential data is today omnipresent and covers an important range of application domains. Sequence pattern mining allows to extract information and knowledge that can be of high added value. However, when the sequence data is rich in numerical data finer data mining methods are required to be able to extract more expressive knowledge representing the variability of numerical values and their possible interdependence. In this paper, we present a new method for the discovery of frequent gradual sequences represented by directed graphs (DAG), from a data sources of sequences in RDF (Resource Description Framework). The former ones, are first transformed into *pog* (partially ordered gradual sequences) DAGs labeled at the arcs and the nodes. Then, on these graphs, we apply an algorithm which mines and discovers the frequent *pog* subgraphs. Experiments on two real datasets in biology have shown the feasibility and the relevance of our approach.