Extraction complète efficace de chemins pondérés dans un a-DAG

Nazha Selmaoui-Folcher*, Frédéric Flouvat*
Chengcheng Mu*, Jérémy Sanhes*, Jean-François Boulicaut**

*PPME - Université de la Nouvelle Calédonie
BP R4, 98851, Nouméa, Nouvelle Calédonie
prenom.nom@univ-nc.nc,
**LIRIS - INSA de LYON
Bâtiment Blaise Pascal F-69621 Villeurbanne cedex, France
jean-francois.boulicaut@insa-lyon.fr

Résumé. Un nouveau domaine de motifs appelé chemins pondérés condensés a été introduit en 2013 lors de la conférence IJCAI. Le contexte de fouille est alors un graphe acyclique orienté (DAG) dont les sommets sont étiquetés par des attributs. Nous avons travaillé à une implémentation efficace de ce type de motifs et nous montrons que l'algorithme proposé était juste mais incomplet. Nous établissons ce résultat d'incomplétude et nous l'expliquons avant de trouver une solution pour réaliser une extraction complète. Nous avons ensuite développé des structures complémentaires pour calculer efficacement tous les chemins pondérés condensés. L'algorithme est amélioré en performance de plusieurs ordres de magnitude sur des jeux de données artificiels et nous l'appliquons à des données réelles pour motiver qualitativement l'usage des chemins pondérés.

1 Introduction

Avec les avancées technologiques en terme d'acquisition des données scientifiques (images satellitaires, capteurs, etc.), les scientifiques s'intéressent de plus en plus à des applications importantes en terme de surveillance et suivi de l'environnement. Les données collectées sont généralement hétérogènes, multiéchelles, spatiales et temporelles (série temporelle d'images satellites, aériennes, modèles numériques de terrain, nature du sol ...) et sont destinées à comprendre et prédire des phénomènes résultant de processus complexes et d'origine pluridisciplinaire (données climatiques, géologiques, ...). L'explosion de cette information spatiale, temporelle et des systèmes d'informations géographiques nécessitent l'investissement dans des méthodes d'extraction de connaissances et nous nous intéressons à celles qui reposent sur la détection de motifs locaux comme, par exemple, la découverte de motifs séquentiels (Agrawal et Srikant, 1995; Mannila et al., 1997; Masseglia et al., 1998) ou de motifs plus complexes comme des sous-graphes Inokuchi et al. (2000) ou des sous-arbres Zaki (2002).
Nos besoins concernent l'étude spatiale et temporelle des évolutions d'objets et de leurs interactions. Les objets peuvent être caractérisés par plusieurs attributs et leurs évolutions que

l'on appelle parfois **dynamique** se décrivent par les évolutions des attributs, par leur emplacement géographique, leur existence (apparition/disparition) et leur structure topologique (fusion/division). Pour certaines applications, nous pouvons transformer la base de données spatio-temporelles dans une base de données transactionnelles Hai et al. (2012) ou dans une base de séquences pour les analyser. Cependant, ces transformations peuvent s'avérer très fastidieuse et les résultats peuvent être difficilement interprétables. Des domaines de motifs plus sophistiqués et applicables à l'étude de phénomènes spatio-temporels ont donc été proposés. Ainsi, plusieurs travaux se sont intéressés à l'extraction de motifs dans des graphes étiquetés Inokuchi et al. (2000). Quelques travaux ont été menés dernièrement sur des graphes attribués (Fukuzaki et al., 2010; Miyoshi et al., 2009; Desmier et al., 2013). Les difficultés dans la fouille de graphes attribués résident dans l'explosion combinatoire de l'exploration de l'espace de recherche. En effet, cette espace de recherche porte à la fois sur les combinaisons de graphes et les combinaisons d'attributs. Dans un travail présenté dans (Sanhes et al. (2013a,b)), Sanhes et al. ont proposé de travailler à la modélisation de données spatio-temporelles dans des DAG attribués, autrement dit un unique graphe orienté acyclique attribué (a-DAG) (cf. Figure 1) : les sommets sont des objets spatiaux caractérisés par un ensemble d'attributs ou caractéristiques et les arcs dénotent la proximité spatio-temporelle entre ces objets (par exemple le voisinage spatial entre deux objets de deux pas de temps consécutifs). Le but est de trouver les transitions ou cheminements de caractéristiques pouvant montrer une tendance attendue ou surprenante, expliquer un phénomène particulier, ce qui revient à chercher dans un a-DAG les chemins fréquents d'attributs. On trouve quelques travaux s'attaquant à la fouille de graphes orientés

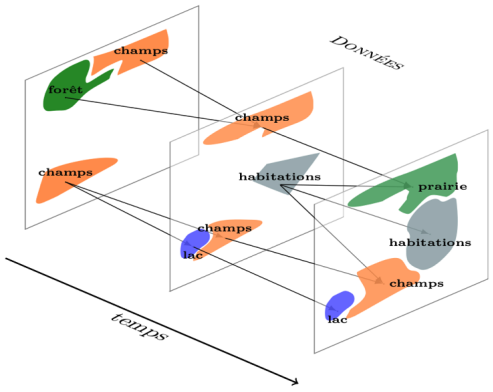


FIGURE 1: Exemple de a-DAG construit sur des objets représentés dans des images temporelles

acycliques mais étiquetés (et non pas attribués) tels que Chen et al. (2004); Termier et al. (2007). Ces méthodes recherchent des sous-graphes dans un ensemble de graphes, et de plus les sommets sont plutôt labélisés ou considérés comme labélisés et non attribués. Dans notre cas, on est en présence d'un seul graphe orienté acyclique et attribué (a-DAG) ce qui pose des problèmes très différents. Le domaine de motif proposé pour la première fois dans Sanhes et al. (2013b) est appelé domaine des **chemins pondérés** dans un a-DAG. Lorsque nous avons voulu travailler à une implémentation efficace de l'algorithme d'extraction de ce type de motif, le seul à notre connaissance qui calcule des motifs dans des DAG attribués, nous avons étudié de près ses propriétés et nous avons découvert qu'il était juste mais incomplet. Nous présentons

ici un nouvel algorithme permettant de réaliser l'extraction de tels motifs de façon complète. Non seulement nous proposons une correction du premier algorithme mais aussi nous étudions des optimisations nécessaires au passage à l'échelle en introduisant des structures de données complémentaires comme un graphe de motifs. Nous montrons que la performance de l'extraction est améliorée de plusieurs ordres de magnitude sur des jeux de données artificiels et nous l'appliquons aussi à des données réelles pour motiver qualitativement l'usage des chemins pondérés.

Dans la section suivante, nous présenterons les concepts et définitions nécessaires à la compréhension de l'algorithme. En Section 3, nous prouvons l'incomplétude de l'algorithme existant. Nous proposons une solution de complétude et une optimisation basée sur une structure de graphe de motifs en Section 4. Nous montrerons les performances de l'algorithme complet et optimisé sur des jeux de données artificielles en Section 5. Et enfin, nous conclurons en Section 6.

2 Chemins pondérés : concepts et définitions

Graphe et chemins Un **DAG attribué** (aussi appelé **a-DAG**) $G = (V, E, \lambda)$ sur un ensemble d'items $\mathcal{I}$ consiste en un ensemble de sommets V, un ensemble d'arêtes orientées ou arcs $E \subseteq V \times V$ et une fonction d'attribution $\lambda : V \rightarrow \mathcal{P}(\mathcal{I})$ qui fait correspondre à chaque sommet du DAG G un sous-ensemble de $\mathcal{I}$.

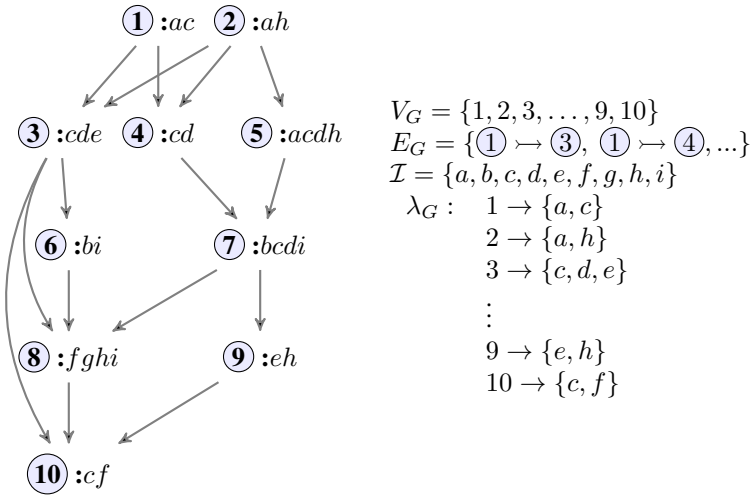


FIGURE 2: Exemple de a-DAG.

Un **chemin** P est une séquence d'itemsets $P = P_1 \rightarrowtail P_2 \rightarrowtail \cdots \rightarrowtail P_n$ tel qu'il existe un chemin $O = v_1 \rightarrowtail v_2 \rightarrowtail \cdots \rightarrowtail v_n$ dans le graphe où P_i est inclus dans l'ensemble des items de v_i (notion à différentier de la définition classique d'un chemin dans un graphe). On dit que alors que O est une **occurrence** du chemin P et l'ensemble des occurrences de P est noté $occu_G(P)$.

Par exemple dans le graphe de la figure 2 les occurrences du chemin de taille 3 $ah \rightarrowtail cd \rightarrowtail i$ sont $\textcircled{2} \rightarrowtail \textcircled{3} \rightarrowtail \textcircled{6}$, $\textcircled{2} \rightarrowtail \textcircled{3} \rightarrowtail \textcircled{8}$, $\textcircled{2} \rightarrowtail \textcircled{4} \rightarrowtail \textcircled{7}$, $\textcircled{2} \rightarrowtail \textcircled{5} \rightarrowtail \textcircled{7}$, et $\textcircled{5} \rightarrowtail \textcircled{7} \rightarrowtail \textcircled{8}$.
Un **chemin pondéré** P est un chemin où un poids est associé à chaque arc $P_i \rightarrowtail P_{i+1}$ constituant P. Ce poids correspond au nombre d'occurrences distinctes de $P_i \rightarrowtail P_{i+1}$ dans le graphe. Pour l'exemple précédent, le chemin $ah \rightarrowtail cd \rightarrowtail i$ et ses occurrences permettent de construire le motif pondéré : $ah \overset{4}{\rightarrowtail} cd \overset{5}{\rightarrowtail} i$. En effet, le nombre d'occurrences de $ah \rightarrowtail cd$ dans $occur_G(P)$ st 4, et le nombre d'occurrences de $cd \rightarrowtail i$ dans $occur_G(P)$ est 5. Une telle représentation permet de voir que l'itemset ah apparaît 4 fois avant l'apparition du chemin $cd \rightarrowtail i$, et que l'itemset i apparaît 5 fois après l'apparition du chemin $ah \rightarrowtail cd$. Dorénavant, $\omega_G(P_i \rightarrowtail P_{i+1})$ désignera le poids de l'arc entre les itemsets P_i et P_{i+1}.

Relation d'inclusion L'opérateur d'inclusion $\sqsubseteq$ sur un couple de chemins pondérés est défini de la manière suivante : $P \sqsubseteq P'$ si et seulement si $|P| \leq |P'|$ et $\exists k \in [0, |P'| - |P|]$ tel que :

$$\begin{cases} \forall i \in [1, |P|], & P_i \subseteq {P'}_{k+i} \qquad \text{(inclusion d'itemsets)} \\ \forall j \in [1, |P|[, & \omega_G(P_j \rightarrowtail P_{j+1}) = \omega_G({P'}_{k+j} \rightarrowtail {P'}_{k+j+1}) \qquad \text{(égalité des poids)} \end{cases}$$

Autrement dit, P est inclus dans P' s'il existe une sous-séquence Q de P' tel que les itemsets de P sont inclus un à un dans ceux de Q avec les mêmes poids au niveau des arcs. On dit que P' est un **sur-chemin** ou **super-chemin** pondéré de P.
A partir de la mesure proposée par Bringmann et Nijssen (2008), nous définissons le **support** d'un chemin pondéré P, noté $\sigma(P)$, comme étant le poids minimal de ses arcs.

Chemin pondéré condensé Un chemin pondéré P est un **condensé** s'il n'admet aucun sur-chemin pondéré.
Pour simplifier, nous appellerons motif un chemin pondéré et les occurrences d'un motif seront tout simplement des chemins.

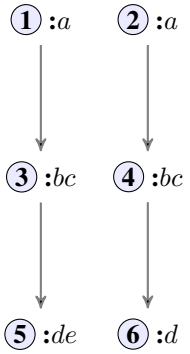
3 Algorithme non complet

L'algorithme d'extraction de chemins condensés a été proposé par J. Sanhes et al. (Sanhes et al. (2013a,b)). Il se déroule en 2 étapes. La première étape concerne l'extraction des motifs condensés de taille 2 appelés **graine** (un seul arc) dans un a-DAG : pour cela les auteurs font remarquer qu'un chemin pondéré $P = P_i \rightarrowtail P_{i+1}$ de taille 2 peut simplement être représenté par un triplet $\{occur_G(P), P_i, P_{i+1}\}$. Ils transforment alors le a-DAG G en une base de données transactionnelles ternaire $(E, (\mathcal{I}), (\mathcal{I}))$ dans laquelle chaque arc $v_i \rightarrowtail v_{i+1}$ est représenté par le tuple $(v_i \rightarrowtail v_{i+1}, \lambda_G(v_i), \lambda_G(v_{i+1}))$. L'extraction est alors réalisée par l'algorithme Data-Peeler [1] développé par (Cerf et al. (2008)) cherchant des ensembles fermés. Les motifs obtenus de taille 2 sont alors des condensés au sens des itemsets caractérisant les sommets d'origine et de destination. La deuxième étape consiste à étendre récursivement chaque graine dans les 2 sens par des itemsets fermés localement dans l'ensemble des itemsets accessibles par un arc. L'algorithme d'extension utilise le concept de base de données projetée.

1. http ://homepages.dcc.ufmg.br/ lcerf/fr/prototypes.html

Cette méthode permet bien l'extraction des motifs condensés de manière juste mais ne les génère pas tous. En effet, toutes les **graines** condensées forment bien des motifs condensés mais un motif condensé de taille supérieure à 2 peut contenir des **graines** non condensées. Effectivement, il existe des motifs condensés au sens de l'inclusion qui peuvent être formés par certains motifs de taille 2 qui ne sont pas générés par la première étape.

Pour illustrer l'incomplétude de l'algorithme, nous montrons un contre-exemple sur le graphe de la figure 3. Dans ce a-DAG, les **graines** générées par la première étape de l'algorithme ne permettent pas de construire le motif condensé $a \overset{1}{\rightarrowtail} bc \overset{1}{\rightarrowtail} de$.



Condensés représentés dans le a-DAG :
- chemins de tailles 2 : $a \overset{2}{\rightarrowtail} bc$, $bc \overset{2}{\rightarrowtail} d$ et $bc \overset{1}{\rightarrowtail} de$
- chemins de tailles 3 : $a \overset{2}{\rightarrowtail} bc \overset{2}{\rightarrowtail} d$ et $a \overset{1}{\rightarrowtail} bc \overset{1}{\rightarrowtail} de$

Graines générées à la première étape :
- $a \overset{2}{\rightarrowtail} bc$ supporté par $①\rightarrowtail③$ et $②\rightarrowtail④$
- $bc \overset{2}{\rightarrowtail} d$ supporté par $③\rightarrowtail⑤$, et $④\rightarrowtail⑥$
- $bc \overset{1}{\rightarrowtail} de$ supporté par $③\rightarrowtail⑤$

FIGURE 3: Exemple de motifs condensés non générés $a \overset{1}{\rightarrowtail} bc \overset{1}{\rightarrowtail} de$.

Ce problème vient du fait qu'à l'extension du mortif P=$a \overset{2}{\rightarrowtail} bc$ on se retrouve dans un cas où l'on perd des occurrences des **graines** à étendre. En effet, pour étendre P=$a \overset{2}{\rightarrowtail} bc$ supporté par $①\rightarrowtail③$ et $②\rightarrowtail④$, on part des sommets représentant l'itemset bc à étendre, i.e. $\{③, ④\}$. Nous pouvons étendre le motif P par les itemsets de et d (itemsets fermés par rapport aux sommets accessibles du motif en question). L'extension par l'itemset d fournit le motif $a \overset{2}{\rightarrowtail} bc \overset{2}{\rightarrowtail} d$ dont les occurrences sont $①\rightarrowtail③\rightarrowtail⑤$, et $②\rightarrowtail④\rightarrowtail⑥$. Dans ce cas toutes les occurrences de P sont utilisées, conservées, c'est une **extension sans perte d'occurrences**. L'extension par l'itemset de fournit le motif $a \overset{\sigma}{\rightarrowtail} bc \overset{1}{\rightarrowtail} de$ où σ ne vaut plus 2 car toutes les occurrences de P ne sont pas utilisées : $②\rightarrowtail④$ n'est pas relié à de. Dans ce cas on parle d'**extension avec perte d'occurrences**. Par conséquent il faut déterminer les occurrences utilisées et mettre à jour les différents poids (σ) ainsi que les itemsets du motif. En réalité avec la séparation des 2 étapes, l'information structurelle est perdue pendant le parcours en profondeur.

4 Algorithme complet et optimisé

Dans ce paragraphe, nous proposerons une solution permettant de corriger la complétude et nous proposerons par la même occasion une version optimisée utilisant une structure de graphe permettant de stocker les motifs que l'on appellera graphe de motifs.

Soit $P = I_1 \overset{\sigma_1}{\rightarrowtail} ... \overset{\sigma_n}{\rightarrowtail} I_n$ un motif. Nous pouvons étendre P **sans perte d'occurrences** s'il existe un itemset I_{n+1} fermé fréquent dans l'ensemble des sommets accessibles par V_n (nous appellerons I_{n+1} un fermé fréquent local à V_n), tels que V_{n+1} supporte I_{n+1} et qu'il existe au moins un arc de chaque sommet de V_n vers V_{n+1}, c'est-à-dire que toutes les occurrences de P sont conservées. De manière analogue, nous pouvons étendre P **avec perte d'occurrences** lorsque toutes les occurrences de P ne sont pas utilisées lors de l'extension. Dans ce cas nous obtenons un motif $P' = I_1 \overset{\sigma'_1}{\rightarrowtail} ... \overset{\sigma'_n}{\rightarrowtail} I_n \overset{\sigma_n}{\rightarrowtail} I_{n+1}$ où les σ'_i sont à mettre à jour.

L'idée de base de notre solution est alors de marquer les extensions avec perte d'occurrences parmi les deux cas identifiés pour y appliquer une étape retour (**backtracking**) et mettre à jour les poids en faisant un parcours inverse du motif et de l'a-DAG.

Si nous reprenons notre exemple de la figure 3, au moment de l'extension du motif P=$a \overset{2}{\rightarrowtail} bc$ avec perte par l'itemset de qui n'est accessible que par le sommet ③ et non le sommet ④, on duplique le motif P par le motif P'=$a \overset{1}{\rightarrowtail} bc$ d'occurrence ①↣③ (cf. figure 3) en mettant à jour les poids de P'.

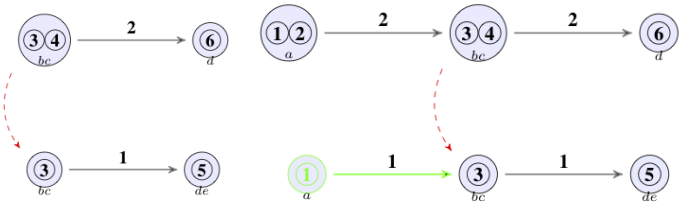


FIGURE 4: Exemple du **backtracking** pour générer $a \overset{1}{\rightarrowtail} bc \overset{1}{\rightarrowtail} de$.

4.1 Stratégie de l'algorithme complet

À partir des notions introduites précédemment, nous pouvons présenter la stratégie générale de l'algorithme complet (cf. algorithme 1). Cet algorithme est basé sur un parcours en profondeur de l'espace de recherche pour étendre les motifs condensés.

En partant de l'ensemble des sommets du graphe, l'algorithme effectue un parcours en profondeur dans l'espace de recherche pour étendre le motif condensé P initialisé à $\emptyset$. La ligne 1 exprime le cas d'arrêt de l'algorithme : l'ensemble des sommets destinations V_P est vide. Le parcours en profondeur se fait aux lignes 2 et 3. L'extension de P se fait avec l'itemset Y fermé fréquent par rapport aux sommets de V_P (ligne 4). Nous notons P' le motif ainsi étendu. Lorsqu'il s'agit d'une extension avec perte d'occurrences, il est nécessaire de mettre à jour les occurrences du motif (ligne 5). Ce nouveau motif P' est potentiellement un motif ou un sous-motif condensé. Nous supprimons les motifs qui sont inclus dans P', et insérons P' dans C l'ensemble des motifs condensés (lignes 7 et 8). Puis nous continuons le parcours (ligne 9).

La complexité de l'algorithme ne permet pas le passage à l'échelle à cause de nombreux tests coûteux d'inclusion de motifs pour vérifier sa maximalité. Nous proposons ci-dessous une implémentation optimisée basée sur une structure de graphe pour stocker les motifs. Cette nouvelle structure va permettre d'éviter les tests trop coûteux de comparaison entre motifs.

Algorithme 1 : DepthFirstMining

Entrées : P motif courant ($\emptyset$ à l'appel initial)
V_P ens. des sommets destinations de P ($V_P = V_G$ à l'appel initial)
C ens. des motifs condensés ($\emptyset$ à l'appel initial)
$G = (V_G, E_G, \lambda_G)$ un a-DAG
min_sup le support minimal

si V_P ! $= \emptyset$ **alors**
  $IFF = MiningClosedItemsets(V_P, minsup)$ // IFF ensemble des itemsets fermés fréquents de V_P.
  pour chaque itemset $Y \in IFF$ **faire**
    Soit P' l'extension de P avec Y.
    si extension avec perte d'occurrences **alors**
      Mettre à jour les occurrences de P'
    Supprimer dans C les motifs Q inclus dans P'
    Insérer P' dans C.
    DepthFirstMining(P', $V_{P'}$, C, G, min_sup)

4.2 Implémentation optimisée

Nous allons nous servir de la maximalité des chemins pondérés condensés recherchés pour optimiser l'algorithme qui se traduit par la maximalité des itemsets (itemsets fermés) du chemin et sa taille. Les tests d'inclusions de l'algorithme se font sur les itemsets et sur les chemins. Pour éviter ces tests nous allons définir une structure de graphe permettant le stockage des motifs trouvés au fur et à mesure du parcours de l'espace de recherche. Cette structure est appelée **graphe de motifs**.

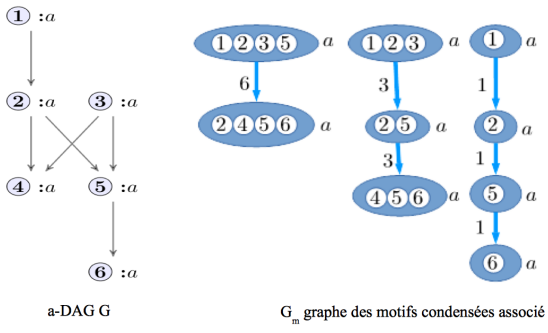


FIGURE 5: Graphe de motifs du a-DAG G : $a \overset{6}{\rightarrowtail} a$, $a \overset{3}{\rightarrowtail} a \overset{3}{\rightarrowtail} a$ et $\overset{1}{\rightarrowtail} a \overset{1}{\rightarrowtail} a \overset{1}{\rightarrowtail} a$

Structure de données pour les chemins pondérés condensés Soit $\mathcal{I}$ un itemset, soit $G = (V, E, \lambda)$ un graphe attribué sur $\mathcal{I}$. Nous modélisons l'ensemble des motifs condensés par un

graphe des motifs défini par $G_m = (V_m, E_m, \lambda_m)$ où :

- $V_m \subseteq \mathcal{P}(V)$ est l'ensemble des sommets
- $E_m \subseteq \mathcal{P}(V) \times \mathcal{P}(V)$ est l'ensemble des arcs
- λ_m : la fonction d'attributs définie par :
  $V_m \longrightarrow \mathcal{P}(I)$
  $V \longmapsto X$ où X représente l'itemset maximal caractérisant les sommets de V.

Réciproquement, on associe à un itemset X l'ensemble des sommets noté V_X supportant l'itemset X. Un sommet du graphe G_m est identifié par un ensemble de sommets du graphe G. Un motif condensé est alors un chemin $c = V_1 \rightarrowtail ... \rightarrowtail V_n$ de G_m de longueur maximale (cf. figure 5), i.e. V_1 est une source (pas d'arc incident) et V_n est un puits (pas d'arc sortant).

Procédure `cherCondRec` (X : itemset, V_X : ens. de sommets, min_sup : entier)

```
1  sommets_a_remonter : var. globale contenant les sommets à backtracker
2  si V_X ≠ ∅ alors
3      E+(V_X) les arcs sortants de V_X, V+(V_X) les sommets sortants de V_X
           BdT(E+(V_X)) la base de données transactionnelles construites à partir de
           E+(V_X) // une transaction est un arc de E+(V_X), les items sont
           ceux du sommet destination de l'arc.
4      pour chaque itemset Y fermé et fréquent dans BdT(E+(V_X)) faire
5          Soit V_Y les sommets supportant Y dans cette base locale
6          Soit V_{X_utile} les sommets de V_X ayant au moins un arc vers V_Y
7          si V_X == V_{X_utile} alors
8              Insérer (V_{X_utile}, X) -> (V_Y, Y) dans le graphe des motifs G_m
9          sinon
10             Insérer V_{X_utile} → V_Y dans le graphe des motifs G_m
11             si le sommet V_X existe dans le graphe des motifs G_m alors
12                 ajouter V_{X_utile}, V_X dans sommets_a_remonter
13         si le sommet V_{X_utile} n'est pas présent dans G_m alors
14             cherCondRec(Y, V_Y)
```

Avec cette structure de graphe de motifs, nous éviterons tous les tests de comparaisons et d'inclusions entre les motifs trouvés. En effet, prenant un motif condensé P= $I_1 \overset{\sigma_1}{\rightarrowtail} ... \overset{\sigma_{n-1}}{\rightarrowtail} I_n$, c'est une séquence d'itemsets et chaque itemsets I_i du motif P représente un sommet dans le graphe des motifs qui n'est autre que V_{I_i} ensemble des sommets du graphe a-DAG contenant l'itemsets I. Le motif P est alors identifié de manière unique par la séquence $V_{I_1} \rightarrowtail ... \rightarrowtail V_{I_n}$ dans le graphe des motifs. Au moment de la construction du graphe des motifs et à l'insertion d'un nouveau sommet V_i dans le graphe des motifs, il suffit de vérifier s'il est déjà présent dans le graphe des motifs alors il a déjà été parcouru sinon il est inséré et le parcours de l'espace de recherche continue. L'algorithme final se déroule en 2 grandes étapes suivantes :
- Recherche des motifs condensés par un parcours en profondeur de l'espace de recherche (procédure 2). Les motifs sont stockés dans le graphe des motifs. Pendant la recherche, les sommets pour lesquels il y a eu extension avec perte d'occurrences sont marqués pour être

traités par la phase de **backtracking**.
- Phase de **backtracking** sur les sommets marqués (pour lesquels il y a eu extension avec perte d'occurrences) pour mettre à jour les occurrences des motifs (cf. procédure 3).
La première étape fait appel à la procédure *cherCondRec* qui effectue un parcours en profondeur de l'espace de recherche. Cette procédure étend récursivement les motifs condensés au fur et à mesure de leur construction dans G_m. A une étape de la construction du motif P, soit V_X le sommet à étendre dans le graphe des motifs G_m supportant l'itemset X, on calcule $V_{X_{utile}}$ ensemble de tous les sommets ayant au moins un arc sortant de V_X. On calcule tous les items accessibles par X, on obtient une base de données transactionnelles dont les transactions sont les arcs sortants et les items sont les items accessibles par X (ligne 3 et 4). Pour chaque itemset maximal Y dans cette base transactionnelle, on va étendre V_X par V_Y. Deux cas se présentent :

- $V_X = V_{X_{utile}}$: tous les sommets de V_X sont utilisés pour l'extension, il y a extension sans perte. Il suffit alors de créer le sommet V_Y et le relier à V_X par un arc dans G_m.
- $V_X \neq V_{X_{utile}}$: l'extension est réalisé avec perte d'occurrences, on duplique le motif P en remplaçant le sommet V_X par $V_{X_{utile}}$ (à marquer pour être traiter dans la phase de **Backtracking**). On insère un nouveau sommet V_Y et on crée un arc entre $V_{X_{utile}}$ et V_Y.

Procédure `backtrackRec` ($V_{X_{utile}}$: sommet du graphe des motifs, $V_{toBacktrack}$: sommet du graphe des motifs, min_sup : entier)

1 **si** $sommets\_a\_remonter$ contient $(V_{toBacktrack}, V_{direction})$ **alors**
  `// i.e.` $V_{toBacktrack}$ `doit être backtracké en suivant` $V_{direction}$
2 $backtrackRec(V_{toBacktrack}, V_{direction})$
3 **pour chaque** V_i sommet incident à $V_{direction}$ dans G_m **faire**
4 Soit $V_{i_{utile}}$ ensemble des sommets de V_i ayant au moins un arc sortant vers $V_{X_{utile}}$.
5 Insérer $V_{i_{utile}} -> V_{X_{utile}}$ dans le graphe des motifs G_m
  `// Lors de l'insertion, mise à jour des attributs.`
6 $backtrackRec(V_{i_{utile}}, V_i)$`// backtracking vers le haut`

L'étape de backtracking décrite par la procédure récursive *backtrackRec* retraite chaque sommet marqué en visitant la branche du motif dans le sens inverse pour mettre à jour les occurrences des motifs et les informations tels que les poids et les itemsets. La figure 6 montre le déroulement de l'algorithme sur l'exemple du graphe de la figure 5. Les arcs du graphe de motifs sont en bleu. Les arcs en rouge définissent le cas d'extension avec perte d'occurrences, les arcs en vert montrent la phase de **backtracking** avec la mise à jour des poids et des itemsets.

5 Expérimentations et résultats

Nous avons appliqué la méthode sur des jeux de données artificielles pour montrer la performance que nous avons comparé avec la méthode incomplète. Pour montrer l'intérêt de ce nouveau domaine de motifs, nous avons utilisé un jeu de données réelles pour le problème de suivi du phénomène de l'érosion.

Dans un premier temps, nous avons créé artificiellement trois jeux de données afin d'observer l'impact de la taille des a-DAG sur les performances notés « V20K, E60K » pour un a-DAG

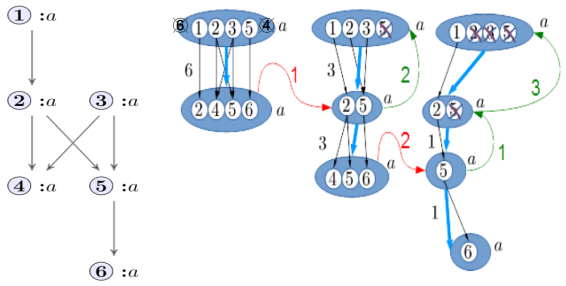


FIGURE 6: Etapes de construction du graphe de motifs du a-DAG à gauche

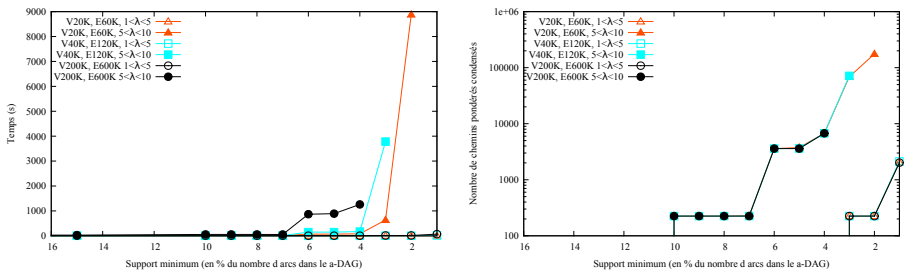


FIGURE 7: Performances et nombre de solutions en fonction des seuils de fréquence (en %).

de 20 000 sommets et 60 000 arêtes, « V40K, E120K » et « V200K, E600K ». Nous avons généré des attributs pour leur sommets. Parmi 15 attributs, et pour chaque sommet, on tire au sort le nombre d'attributs, puis les attributs eux-mêmes. On a pour chaque jeu de données, une version dont le nombre d'attributs varie entre 1 et 5 items, et une version où le nombre varie entre 5 et 10 items. Les temps d'exécution et le nombre de chemins pondérés fréquents sont reportés sur la figure 7. Dans un deuxième temps, nous avons voulu nous assurer que le jeu de

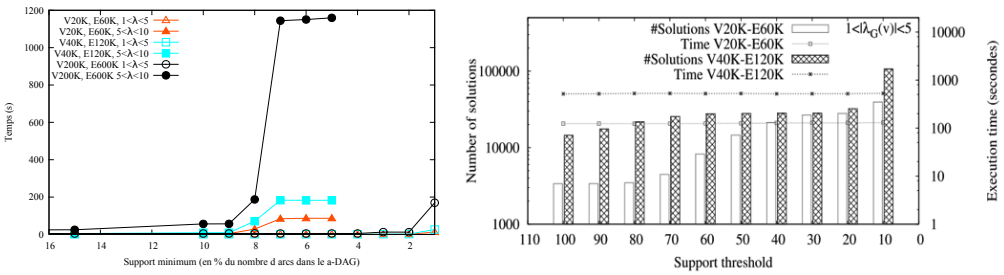
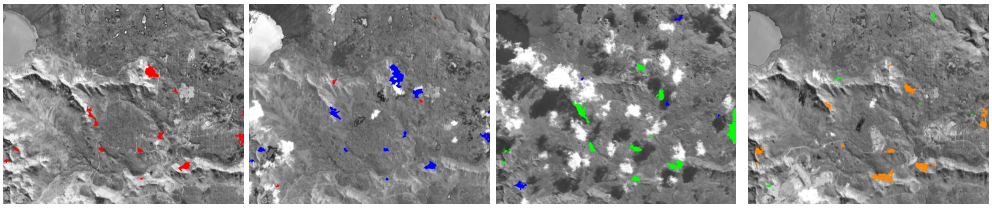


FIGURE 8: Performances des deux algorithmes pour les données artificiels (5 couches).

données artificiel ressemble plus aux jeux de données tirés d'une application spatio-temporelle.

En l'occurrence, ceux-ci comportent une répartition par niveau sur les a-DAG ; chaque niveau correspond à une tranche temporelle, et il n'existe d'arc qu'entre deux tranches consécutives. Nous avons donc réparti en 5 niveaux les a-DAG générés précédemment. Les résultats sont reportés sur la figure 8 (les courbes de droite). Le graphe de gauche sur la figure 8 regroupe les résultats en temps d'exécution et en nombre de solutions trouvés par l'algorithme incomplet et non optimisé. Les courbes montrent une nette optimisation en temps de calculs puisque sur le même jeu de données le temps a été réduit d'un facteur de 1000. Le stockage des motifs dans une structure de graphe de motifs a permis également de réduire la taille de la mémoire.
Nous disposons des images satellites de résolution 10m à des dates non régulières. Nous avons travaillé sur une zone couvrant une variété de régions observables : points d'eau douce ou marine, activité anthropiques (mines, usines et pistes), relief, bassins versants, plaines, forêts. Avant l'analyse des évolutions, les images ont été segmentées en régions. Les attributs associés à ces régions sont (Redness : rougeur, NDVI : indice de végétation, Brightness : indice de brillance et Slope : pente). Ces attributs ont été discrétisés en 5 classes (Redness0, ..., Redness4) de faibles valeurs au plus fortes. La figure 9 illustre le motif $Redness1 \overset{20}{\rightarrowtail} Redness1 \overset{21}{\rightarrowtail}$



1999 2002 2005 2008

FIGURE 9: Redness1 $\overset{20}{\rightarrowtail}$ Redness1 $\overset{21}{\rightarrowtail}$ Redness2, Slope $= [3{,}6\,;\,30] \overset{23}{\rightarrowtail}$ Redness3 .

$Redness2, Slope = [3{,}6\,;\,30] \overset{23}{\rightarrowtail} Redness3$ avec une augmentation progressive du Redness qui témoigne d'une érosion naturelle. La valeur Slope=[3,6 ; 30] indique que ces zones possèdent des pentes entre $3,6\%$ et 30%. En effet ces zones correspondent aux versants en bas des crêtes de massifs ou à des bords de pistes (risque d'érosion fort).

6 Conclusion

Nous avons présenté une solution permettant de corriger l'algorithme d'extraction de chemins pondérés condensés développé en 2013 en proposant un algorithme efficace et complet. L'optimisation de la nouvelle version repose sur une structure de graphes qui sert de structure de stockage des motifs trouvées. Nous avons testé les performances sur des jeux de données synthétiques en les comparant aux performances de l'ancien algorithme. Les résultats montrent une nette réduction en temps d'exécution et en mémoire. L'algorithme optimisé est complet et passe à l'échelle. Les chemins pondérés condensés sont intéressants pour analyser les évolutions spatio-temporelles d'objets. Comme le montrent les résultats qualitatifs sur les données réelles liées au problème de l'érosion. Les motifs trouvés permettent de suivre les évolutions des régions en ayant des informations sur les évolutions des caractéristiques et sur la structure de l'objet grâce à la structure de graphe de motifs.

Références

Agrawal, R. et R. Srikant (1995). Mining sequential patterns. In *ICDE'95*, pp. 3–14.

Bringmann, B. et S. Nijssen (2008). What is frequent in a single graph ? In *PAKDD'08*, pp. 858–863.

Cerf, L., J. Besson, C. Robardet, et J. Boulicaut (2008). Data peeler : Contraint-based closed pattern mining in n-ary relations. In *SIAM/SDM'08*, pp. 37–48.

Chen, Y., H. Kao, et M. Ko (2004). Mining DAG patterns from DAG databases. In *WAIM'04*, pp. 579–588.

Desmier, E., M. Plantevit, C. Robardet, et J.-F. Boulicaut (2013). Trend Mining in Dynamic Attributed Graphs. In *ECML/PKDD'13*, pp. 654–669.

Fukuzaki, M., M. Seki, H. Kashima, et J. Sese (2010). Finding itemset-sharing patterns in a large itemset-associated graph. In *PAKDD '10*, pp. 147–159.

Hai, P., D. Ienco, P. Poncelet, et M. Teisseire (2012). Extracting trajectories through an efficient and unifying spatio-temporal pattern mining system. In *ECML/PKDD'12*, pp. 820–823.

Inokuchi, A., T. Washio, et H. Motoda (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD'00*, pp. 13–23.

Mannila, H., H. Toivonen, et A. I. Verkamo (1997). Discovery of frequent episodes in event sequences. *DAMI. 1*(3), 259–289.

Masseglia, F., F. Cathala, et P. Poncelet (1998). The psp approach for mining sequential patterns. In *PKDD'98*, pp. 176–184.

Miyoshi, Y., T. Ozaki, et T. Ohkawa (2009). Frequent pattern discovery from a single graph with quantitative itemsets. In *IEEE ICDM Workshops 2009*, pp. 527–532.

Sanhes, J., F. Flouvat, C. Pasquier, N. Selmaoui, et J. Boulicaut (2013a). Extraction de motifs condensés dans un unique graphe orienté acyclique attribué. In *EGC'13*, pp. 205–216.

Sanhes, J., F. Flouvat, C. Pasquier, N. Selmaoui-Folcher, et J. Boulicaut (2013b). Weighted path as a condensed pattern in a single attributed DAG. In *IJCAI'13*, pp. 1642–1648.

Termier, A., Y. Tamada, K. Numata, S. Imoto, T. Washio, et T. Higuchi (2007). Digdag, a first algorithm to mine closed frequent embedded sub-dags. In *MLG 2007*.

Zaki, M. J. (2002). Efficiently mining frequent trees in a forest. In *KDD'02*, pp. 71–80.

Summary

New pattern domain of weighted paths has been introduced in IJCAI 2013 proceedings. The data is modeled as a directed acyclic graph whose nodes are described by means of attributes. We decided to design an efficient and scalable implementation of the proposed algorithm for weighted path computation. Our first result is that we can prove and explain that the first algorithm by Sanhes et al. is correct but incomplete w.r.t. the pattern domain specification. Then, we design an efficient algorithm that is also complete. Its performances are several orders of magnitude better than the pioneer implementation. We apply our algorithm on real environmental data to discuss the qualitative added-value of this promising pattern domain.