Extraction de motifs condensés dans un unique graphe orienté acyclique attribué

Jérémy Sanhes*, Frédéric Flouvat*, Claude Pasquier*,**
Nazha Selmaoui*, Jean-François Boulicaut***

*Université de Nouvelle Calédonie, PPME, BP R4, F-98851 Nouméa, Nouvelle Calédonie
prenom.nom@univ-nc.nc
**Institut de Biologie Valrose (IBV)
UNS - CNRS UMR7277 - INSERM U1091, F-06108 Nice cedex 2
http://ibv.unice.fr claude.pasquier@unice.fr
***Université de Lyon, CNRS, INSA-Lyon, LIRIS UMR5205, F-69621, France
jean-francois.boulicaut@insa-lyon.fr

Résumé. Les graphes orientés acycliques attribués peuvent être utilisés dans beaucoup de domaines applicatif. Dans ce papier, nous étudions un nouveau domaine de motif pour permettre leur analyse : les chemins pondérés fréquents. Nous proposons en conséquence des contraintes primitives permettant d'évaluer leur pertinence (par exemple, les contraintes de fréquence et de compacité), et un algorithme extrayant ces solutions. Nous aboutissons à une représentation condensée dont l'efficacité et le passage à l'échelle sont étudiés empiriquement.

1 Introduction et état de l'art

Les graphes sont omniprésents dans divers domaines d'analyse de données. Récemment, des modèles de graphes plus riches ont été considérés où, par exemple, les sommets et les arêtes sont étiquetés par plusieurs attributs ou propriétés (les itemsets), au lieu d'un seul attribut. Par exemple, un réseau social peut être représenté par un grand graphe où chaque sommet désignerait une personne ainsi que ses domaines d'intérêt. Ces graphes sont des graphes attribués, tels que décrits dans Fukuzaki et al. (2010). Assez souvent, par exemple quand la notion de temps est exprimée, les arêtes sont orientées et les graphes sont acycliques. Nous nous sommes entre autres intéressés à l'analyse de données spatio-temporelles qui peuvent être modélisés via un graphe orienté acyclique attribué (a-dag). Dans un a-dag, les sommets peuvent représenter des objets spatiaux, caractérisés par un ensemble d'attributs ou événements, tandis que les arcs peuvent exprimer leur proximité spatio-temporelle (c-à-d, les objets voisins dans des temps consécutifs).

Dans un a-dag représentant la propagation d'une maladie vectorielle (p.ex. la Dengue) dans une ville, les sommets représenterait des quartiers de la ville à un temps donné. Ils seraient caractérisés par un ensemble d'attributs ou d'événements. Les arcs exprimeraient la propagation de la maladie de quartiers en quartiers à des temps successifs. Dans le cas de l'étude de l'érosion des sols, les nœuds du a-dag seraient des objets géologiques (p.ex. ravine) observés à un

temps donné. Les itemsets décrivent les caractéristiques de ces objets. Les arêtes symboliseraient des relations de fusion ou de division (p.ex. fusion de plusieurs zones érodées).

Nous abordons dans cet article le problème de l'extraction de chemins pondérés fréquents dans un unique a-dag, où chaque poids exprime la fréquence d'une transition. Les chemins fréquents sont utilisés pour analyser la relation causale entre séquences d'événements et/ou attributs. Comme le nombre de motifs peut être très grand, nous avons conçu une représentation condensée pour de telles collections. Ces travaux se situent à l'intersection de plusieurs sujets importants tels que l'extraction de séquences et la fouille de graphes.

Plusieurs algorithmes ont été proposés pour l'extraction de séquences fréquentes. La fermeture, qui a été intensivement étudiée pour la fouille d'itemsets (Pasquier et al., 1999) a été étendue à la fouille de séquences (Yan et al., 2003). Un motif est fermé s'il n'existe aucun super-motif (selon une relation de généralisation/spécialisation) apparaissant dans les mêmes transactions (et donc ayant le même support). De la même manière, plusieurs algorithmes ont été proposés pour extraire les motifs dans des transactions de graphes (Inokuchi et al. (2000), Borgelt et Berthold (2002), Washio et Motoda (2003), Wang et al. (2006)). Les représentations condensées de sous-graphes fréquents ont été étudiées dans Yan et Han (2003) et Termier et al. (2007). Yan et Han (2003) proposent d'extraire les sous-graphes fermés dans une base de données transactionnelle, et Termier et al. (2007) adaptent cette approche pour les DAGs. Ces études se concentrent sur la recherche de motifs fréquents dans des transactions. Dans nos travaux, nous voulons considérer le cas d'un unique (grand) DAG, et l'extraction dans une telle structure soulève différents problèmes.

Comme mentionné par Kuramochi et Karypis (2005), même si les bases de données sous forme d'un unique graphe ou de plusieurs transactions de graphes partagent certaines propriétés, les algorithmes développés pour ces dernières ne peuvent être utilisés dans le cas d'un seul graphe, bien que l'inverse soit vrai. L'un des premiers problèmes est de savoir comment définir la fréquence d'un motif. En effet, cela ne peut pas être le nombre de transactions dans lesquelles le motif apparaît. Plusieurs articles ont étudié ce problème (Kuramochi et Karypis (2005), Fiedler et Borgelt (2007), Bringmann et Nijssen (2008)). La plupart définissent une fréquence qui se fonde sur les occurrences, ce qui n'est pas si simple : plusieurs occurrences peuvent se chevaucher, menant à une mesure qui n'est pas anti-monotone. Sur ces définitions de fréquence, plusieurs algorithmes ont été proposés pour extraire des motifs fréquents dans un unique graphe (Cook et Holder (1994), Matsuda et al. (2000), Kuramochi et Karypis (2005), Gudes et al. (2006)). D'autre part, le nombre de motifs fréquents peut être énorme. Il apparaît donc judicieux de rechercher une représentation condensée, telle que l'ensemble des fermés. Ceci n'est cependant pas trivial puisque les fermés sont généralement définis par rapport à une connexion de Galois entre motifs et transactions.

De plus, la plupart des algorithmes d'extraction de graphes s'appliquent sur des graphes étiquetés, où chaque sommet ou arc n'a qu'une seule étiquette associée. Fouiller des graphes attribués mène à une explosion combinatoire : l'espace de recherche porte à la fois sur les graphes et les attributs. Peu d'études ont considéré les graphes attribués. Miyoshi et al. (2009) fouillent un graphe étiqueté attribué, à savoir un graphe avec des étiquettes et des itemsets quantitatifs dans les sommets. En gardant les étiquettes, la recherche de motifs fréquents est simplifiée et décomposée en deux étapes : extraction de graphes étiquetés puis d'itemsets. Moser et al. (2009) ainsi que Fukuzaki et al. (2010) se concentrent sur la fouille de motifs cohésifs et de motifs partageant des itemsets, c'est-à-dire des motifs représentant des sous-

graphes avec des attributs partagés. A notre connaissance, extraire des motifs fréquents dans un unique graphe attribué n'a pas encore été étudié.

L'identification de motifs fréquents a déjà été étudiée dans le contexte de fouille de graphes. Par exemple, Chen et al. (1998) extraient des motifs fréquents de parcours de chemins dans un graphe orienté acyclique étiqueté représentant les accès utilisateur à travers des pages web. Les parcours de chemins ont aussi été étudiés par Borges et Levene (2000), Nanopoulos et Manolopoulos (2001), Geng et al. (2007). Cependant, dans leurs travaux, ils ne considèrent que des graphes étiquetés, et non des graphes attribués. Par ailleurs, leur définition de chemin est différente de celle que nous utilisons : ils ajoutent de fortes contraintes comme l'évitement de répétitions de sommets et/ou d'arcs.

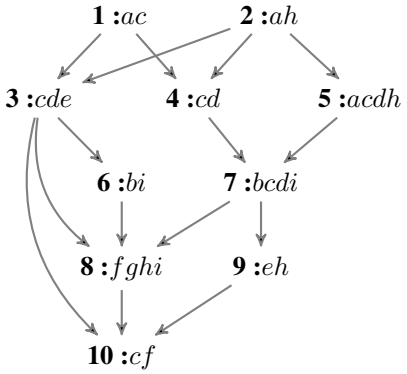
Notre contribution est double. Tout d'abord, nous proposons une première définition de ce qui pourrait être une représentation condensée (exacte) dans le cas d'un unique graphe (voir section 3). Deuxièmement, nous proposons un algorithme efficace pour extraire des motifs fréquents (les chemins pondérés) dans un unique a-dag (voir section 4). Nos expérimentations soulignent les bonnes performances de notre algorithme ainsi que le très fort taux de compression fourni par la représentation condensée (voir section 5).

2 Définitions préliminaires et notations

Dans cette section, nous présentons certains concepts et définitions sur les graphes dirigés acycliques attribués et les chemins pondérés.

2.1 Graphes orientés acycliques attribués

DAG attribué. Un DAG (graphe orienté acyclique) attribué noté $G = (V_G, E_G, \lambda_G)$ sur un ensemble d'items $\mathcal{I}$, aussi appelé **a-dag**, consiste en un ensemble de sommets V_G, un ensemble d'arcs (orientés) $E_G \subseteq V_G \times V_G$ et une fonction d'étiquetage $\lambda_G : V_G \to \mathcal{P}(\mathcal{I})$ qui associe à chaque sommet du DAG G un sous-ensemble de $\mathcal{I}$ [1]. Un exemple est donné en figure 1.



$$
\begin{aligned}
&V_G = \{1, 2, 3, \ldots, 9, 10\}\\
&E_G = \{1 \rightarrowtail 3,\ 1 \rightarrowtail 4,\ 2 \rightarrowtail 3, \ldots\}\\
&\mathcal{I} = \{a, b, c, d, e, f, g, h, i\}\\
&\lambda_G : \quad 1 \to \{a, c\}\\
&\qquad\quad 2 \to \{a, h\}\\
&\qquad\quad 3 \to \{c, d, e\}\\
&\qquad\quad \vdots\\
&\qquad\quad 9 \to \{e, h\}\\
&\qquad\quad 10 \to \{c, f\}
\end{aligned}
$$

FIG. 1: *Exemple de a-dag.*

1. La notation $\mathcal{P}(\mathcal{I})$ désigne l'ensemble des parties $\mathcal{I}$

Chemin et occurrence de chemin. Soit P une suite d'itemsets $I_i \in \mathcal{P}(\mathcal{I})$ notée $P = I_1 \rightarrowtail I_2 \rightarrowtail \cdots \rightarrowtail I_{|P|}$. P est appelé **chemin** ssi il existe une suite de sommets $v_1, v_2, \ldots, v_{|P|} \in V_G$ qui satisfont $\forall I_i \in P$, $I_i \subseteq \lambda_G(v_i)$, et chaque sommet v_i est un père de v_{i+1} dans G. La succession de sommets $O = v_1 \rightarrowtail v_2 \rightarrowtail \cdots \rightarrowtail v_{|P|}$ est une **occurrence** de P. Par exemple, dans le a-dag donné en figure 1, les occurrences du chemin de **taille 3** $ah \rightarrowtail cd \rightarrowtail i$ sont $2 \rightarrowtail 3 \rightarrowtail 6$, $2 \rightarrowtail 3 \rightarrowtail 8$, $2 \rightarrowtail 4 \rightarrowtail 7$, $2 \rightarrowtail 5 \rightarrowtail 7$ et $5 \rightarrowtail 7 \rightarrowtail 8$. L'occurrence $2 \rightarrowtail 3 \rightarrowtail 6$ quant à elle **supporte** les chemins $ah \rightarrowtail cde \rightarrowtail bi$, $a \rightarrowtail cde \rightarrowtail bi$, $h \rightarrowtail cde \rightarrowtail bi$, $h \rightarrowtail c \rightarrowtail bi$, etc.

De plus, nous supposerons que P_i représente le $i^{\text{ème}}$ itemset de P, que O_i représente le $i^{\text{ème}}$ sommet de O, et que $occur_G(P)$ représente l'ensemble des occurrences de P dans G.

2.2 Chemins pondérés

Un chemin classique décrit correctement une séquence d'événements dans un a-dag. Cependant, il serait utile de connaître la contribution de chaque arc dans les occurrences du motif. Nous proposons donc un nouveau domaine de motif : les chemins pondérés.

Chemin pondéré. Les chemins pondérés sont des chemins avec un poids sur chaque arc, représentant le nombre d'occurrences différentes de cet arc parmi toutes les occurrences dans le chemin. Dans les données de l'exemple de la figure 1, le chemin $P = ah \rightarrowtail cd \rightarrowtail i$, dont les occurrences ont été énumérées précédemment, nous donne le motif :

$$ah \overset{4}{\rightarrowtail} cd \overset{5}{\rightarrowtail} i.$$

En effet, les différentes occurrences de $ah \rightarrowtail cd$ dans $occur_G(P)$ sont au nombre de 4, et les différentes occurrences de $cd \rightarrowtail i$ dans $occur_G(P)$ sont au nombre de 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 $\sqsubseteq$ sur un couple de chemins pondérés est défini de la manière suivante : $P \sqsubseteq P'$ ssi $|P| \leq |P'|$ et $\exists k \in [0, |P'| - |P|[$ tel que

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

Un chemin pondéré P' absorbe un autre chemin pondéré P si nous pouvons trouver P dans une sous-séquence de P'. Nous disons ainsi que P' est un **super-chemin pondéré** de P, ou que P' **contient** P.

2.3 Définition du problème

Un des problèmes populaires en fouille de données est la recherche de motifs fréquents, où il s'agit de trouver des motifs dont le support/la fréquence est supérieur(e) à un seuil donné. A

partir de la mesure proposée par Bringmann et Nijssen (2008) [2], nous définissons une mesure de support anti-monotone pour un motif P dans un a-dag G, notée $\sigma_G(P)$:

$$\begin{aligned}\sigma_G(P) &= \min_{1\leq i<|P|} |\{O_i \rightarrowtail O_{i+1} \,/\, O \in occur_G(P)\}| \\ &= \min_{1\leq i<|P|} \omega_G(P_i \rightarrowtail P_{i+1})\end{aligned}$$

La collection de motifs fréquents dans G est l'ensemble des motifs P tels que $\sigma_G(P) > minsup$, $minsup$ étant un seuil défini par l'utilisateur. Cependant, le nombre de chemins fréquents dans G peut être très grand. Dans une telle situation, il peut être judicieux de se pencher vers une représentation condensée des chemins fréquents.

Dans un a-dag G donné, nous cherchons une représentation condensée notée $cond(G)$ de tous les chemins pondérés : chaque chemin pondéré fréquent *ainsi que son support* doit être déductible de $cond(G)$. De plus, aucun élément de $cond(G)$ ne doit pouvoir être déductible à partir d'un autre élément de $cond(G)$. En d'autres termes, nous avons besoin à la fois de la maximalité, de l'unicité et de la complétude des solutions dans la collection $cond(G)$.

3 Représentation condensée exacte des chemins fréquents

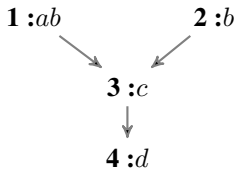
Plusieurs chercheurs se sont penchés sur l'extraction de motifs fermés (voir, par exemple, Pasquier et al. (1999), Yan et al. (2003), Yan et Han (2003)). Ces motifs fermés (fréquents) forment une représentation condensée exacte des motifs fréquents : leur nombre est beaucoup plus petit, bien qu'il soit possible d'en déduire l'ensemble des motifs fréquents.

Fermés et condensés Clarifions tout d'abord les différences de représentation condensée dans un seul graphe et dans une base de données transactionnelle contenant des graphes. Dans cette dernière, la forme la plus commune de représentation condensée est l'ensemble des motifs (itemsets, arbres ou graphes, etc.) fermés, dont la définition se base sur une connexion de Galois entre transactions et motifs. Une propriété importante de l'opérateur de fermeture dans ce type de données est la préservation du support (les motifs et leur fermé ont le même support). Cette propriété repose sur la définition du support : une transaction contenant une ou plusieurs occurrences d'un motif n'est comptée qu'une fois. Dans le contexte d'un graphe unique, la définition de support est très différente. De plus, un motif ne peut avoir qu'un seul fermé dans le cadre des bases de données transactionnelles. Dans notre contexte, cette propriété n'est plus valide : un chemin pondéré peut être déduit à partir de plusieurs super-chemins pondérés (cf exemple en figure 2). La notion de fermeture peut donc difficilement être adaptée dans notre cas.

Une représentation condensée des chemins Comme il est possible de directement lire directement le support d'un chemin pondéré à partir de celui-ci (en prenant le poids minimum parmi tous ceux de notre motif), nous pouvons définir une représentation condensée comme suit :

$$cond(G) = \{P \,/\, \nexists P' \text{ t.q. } P \sqsubseteq P'\}$$

2. Cette mesure peut tout aussi bien s'appliquer sur des sommets ou des arcs



chemin pondéré	condensé candidat
$c \overset{1}{\rightarrowtail} d$	$ab \overset{1}{\rightarrowtail} c \overset{1}{\rightarrowtail} d$
	$b \overset{2}{\rightarrowtail} c \overset{1}{\rightarrowtail} d$

FIG. 2: *Un chemin peut être inclus dans plusieurs chemins condensés.*

Notons qu'ici nous n'avons pas besoin de vérifier l'égalité du support, puisque cette information est déjà déductible du motif lui-même.

Théorème 3.1 - Chaque motif solution de G ainsi que son support peut être déduit de $cond(G)$.

Démonstration. a) D'après les définitions des chemins pondérés et de σ_G, le support d'un chemin pondéré en est directement déductible (minimum des poids). Il suffit donc de lire un chemin pondéré pour avoir son support. *b*) La définition de la relation d'inclusion conserve à la fois l'information sur les attributs ainsi que le poids des arcs ; un chemin pondéré peut donc être déduit à partir de n'importe lequel de ses super-chemins pondérés. □

4 Extraction des chemins pondérés condensés

Nous proposons un algorithme en deux étapes pour extraire les chemins pondérés condensés. Tout d'abord, nous extrayons tous les chemins pondérés de taille 2 (avec un seul arc) qui sont condensés par rapport à l'ensemble des chemins pondérés de taille 2 seulement. Puis, nous effectuons une recherche en profondeur pour obtenir l'ensemble des condensés, en étendant chaque motif de taille 2 précédemment trouvé.

4.1 Extraction des chemins pondérés de taille 2

Les chemins pondérés de taille 2 se représentent naturellement par des triplets de la forme $\{\omega_G(P_{orig} \rightarrowtail P_{dest}), I_{orig}, I_{dest}\}$. Ces triplets correspondent à des concepts triadiques fréquents tels que définis par Cerf et al. (2008). Nous utilisons donc leur algorithme Data-Peeler pour calculer les chemins pondérés de taille 2.

Proposition 4.1 - Soient $L2W$ l'ensemble des chemins pondérés de taille 2, ainsi que la relation ternaire suivante :

1. La première dimension est E_G, la seconde et la troisième sont $\mathcal{I}$,
2. Pour un arc $v_1 \rightarrowtail v_2 \in E_G$ donné, le tuple correspondant $T \in (E_G, \mathcal{P}(\mathcal{I}), \mathcal{P}(\mathcal{I}))$ est $T = (v_1 \rightarrowtail v_2, \lambda_G(v_1), \lambda_G(v_2))$,

Les motifs fermés de dimension 3 (c-à-d, les chemins de taille 2) appelés $LC2W$ **sont équivalents aux** chemins pondérés condensés par rapport à $L2W$.

Démonstration. Reprenons la définition d'un fermé de dimension 3 d'après Cerf et al. (2008) : un motif de dimension 3 S est fermé ssi il n'existe aucun autre motif de dimension 3 S' tel que $\forall i \in \{1, 2, 3\}, S^i \subseteq S'^i$. Comme les dimensions 2 et 3 représentent respectivement les

itemsets des nœuds d'origine et de destination, la définition des motifs fermés sur $(E_G, \mathcal{I}, \mathcal{I})$ est la définition des chemins pondérés condensés de taille 2 (mais condensés seulement par rapport aux chemins pondérés *de taille 2*). Leur poids est le nombre d'arcs différents, qui est en fait la taille de S^1. □

Par exemple, à partir du motif de dimension 3 $S = \{\{1\rightarrowtail 3, 1\rightarrowtail 4\}, \{b, c\}\{c, d, e\}\}$, on obtient le chemin pondéré $P = bc \overset{2}{\rightarrowtail} cde$. Avoir un tel ensemble de chemins pondérés de taille 2 nous permet de procéder à une extension via une recherche en profondeur, à partir de n'importe quel chemin pondéré de taille 2 trouvé.

4.2 Extension des chemins pondérés

Le but de la deuxième étape est d'étendre les chemins pondérés jusqu'à ce qu'ils deviennent des condensés. Pour ce faire, nous exploitons les chemins pondérés précédemment extraits qui sont assurés d'être maximaux au niveau des itemsets des nœuds d'origine et de destination. Leur extension doit être effectuée à la fois vers le bas (en ajoutant des fils) et le haut (en ajoutant des pères). Un nœud dans un a-dag peut avoir plusieurs fils et pères. Pour gérer toutes les combinaisons possible d'extension, nous utilisons deux arbres préfixés (un pour chacune des directions d'extension), tel que montré dans la figure 3. Par manque d'espace, nous ne présentons que l'extension vers le bas. Celle vers le haut est faite en suivant les mêmes principes (en considérant simplement les sommets pères au lieu des sommets fils).

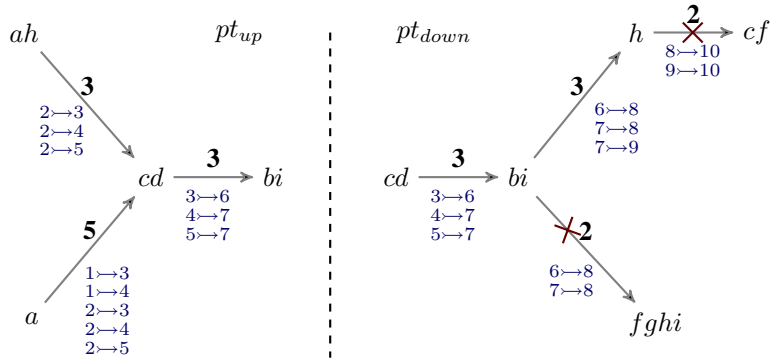


FIG. 3: *Arbres préfixés pour les extensions de* $cd \overset{3}{\rightarrowtail} bi$ *(figure 1,* $minsup = 3$*).*

Pour un motif P, nous définissons $V_{dest} = \{v_{|P|} \in occur_G(P)\}$ et $Li = \{i \in \mathcal{I}$, tel que $\forall v \in V_{dest}, v$ a au moins un fils u t.q. $i \in \lambda_G(u)\}$. Li est la liste des items qui sont dans au moins un fils de chaque sommet destination de P. La méthode d'extension est fondée sur la proposition suivante, dérivée de la proposition 4.1 :

Proposition 4.2 - Soit P un chemin pondéré à étendre (vers le bas). Soit $DB|_P$, la relation binaire projetée par rapport à P, définie comme suit :

1. La première dimension est E_G, la seconde est Li,

2. Pour un arc $v_1 \rightarrowtail v_2 \in E_G$ tel que $v_1 \in V_{dest}$, la transaction correspondante $T \in (E_G, \mathcal{P}(Li))$ est $T = (v_1 \rightarrowtail v_2, \lambda_G(v_2) \cap Li)$

Les chemins pondérés étendus $\{P \rightarrowtail u \mid u$ est fermé dans $DB|_P\}$ sont des super-chemins pondérés de P. Ainsi, il suffit d'étendre récursivement les super-chemins pondérés pour en obtenir le(s) condensé(s).

Par manque de place, nous ne donnerons qu'une intuition rapide de la preuve. Premièrement, l'extension proposée garantit l'inclusion des itemsets et la préservation des poids. Suite à l'extension, nous obtenons donc un super-chemin pondéré de P. Si celui-ci ne peut plus être étendu, alors il est condensé. En effet, il ne peut pas exister de sur-motifs ayant les mêmes poids (de part la définition de fermés dans $DB|_P$, chaque itemset étant un fermé obtenu par projections successives de la base). Les derniers super-chemins pondérés générés ainsi récursivement sont donc des condensés.

Algorithme Chaque chemin pondéré de taille 2 trouvé précédemment (qui peut ne pas être condensé) est récursivement étendu jusqu'à ce qu'il devienne condensé (algo.1 lignes 3-8) [3]. Pour étendre un chemin P, nous regardons V_{dest}, les sommets de destination de $occur_G(P)$ (les derniers sommets de $occur_G(P)$). Si un chemin P peut être étendu avec un itemset I, alors chaque sommet destination de $occur_G(P)$ doit avoir au moins un fils dont l'itemset associé inclut I. Cet ensemble d'itemsets obtenu à partir de $occur_G(P)$ constitue la base de données projetée de P ($DB|_P$, algo.2 ligne 2).

A partir de cette projection, nous étendons le chemin avec les itemsets satisfaisant la définition de la représentation condensée. Pour ce faire, nous extrayons les itemsets fermés dans la base de données projetée $DB|_P$ (algo.2, ligne 5), tel qu'expliqué dans la proposition 4.2. L'extension est alors effectuée (algo.2, lignes 6-12) pour chaque itemset généré vérifiant la contrainte de support. Si l'un des chemins étendus absorbe un chemin pondéré de taille 2 (algo.2 ligne 9), ce dernier ne subira pas d'extension dans les itérations suivantes (algo. 1 lignes 2,3). Nous utilisons pour cela un simple marquage booléen (algo.1 lignes 1,3,4 et algo.2 ligne 10).

Algorithme 1: *Traiter chaque chemin pondéré de taille 2.*

Entrée : $LC2W$

```
1 Initialisation : aucun élément de LC2W n'est marqué
2 pour chaque c2w ∈ LC2W faire
3     si c2w n'est pas marqué alors
4         Marquer c2w
5         Créer deux arbres préfixés pt_down et pt_up
6         EtendreChemin (G,LC2W,c2w,pt_down)          // vers le bas
7         EtendreChemin (G^-1,LC2W^-1,c2w^-1,pt_up)   // vers le haut
8         Générer les solutions à partir de pt_down et pt_up
```

3. Par souci de lecture, l'exposant $\cdot^{-1}$ appliqué à un ensemble d'arcs signifie que nous avons inversé leur direction. Ceci est seulement utilisé pour désigner une extension vers les pères (vers le haut) au lieu des fils.

Algorithme 2: *EtendreChemin.*

Entrées : $LC2W$, le a-dag G, le chemin pondéré P candidat à l'extension et pt l'arbre préfixé
Sorties : $LC2W$, pt

1 $V_{dest} := \{$sommets destination de $occur_G(P)\}$
2 **si** $\exists v \in V_{dest}$ tel que v n'a pas de fils **alors arrêter**
3 $Li := \{i \in \mathcal{I}$, tels que $\forall v \in V_{dest}, v$ a au moins un fils u telles que $i \in \lambda_G(u)\}$
`// Li est la liste des items qui sont dans au moins un fils de chaque sommet destination de P`
4 $DB|_P := \{$transactions $T = \{v \rightarrowtail u,\ i_1 i_2 \dots i_N\}$ t.q. $i_k \in LI \cap \lambda_G(u)\}$
5 $LCI := \{$itemsets fermés CI de $DB|_P$ tels que $\sigma(CI) \geq minsup\}$
`// Le support de chaque CI est le nombre d'arcs`
6 **pour chaque** $CI \in LCI$ **faire**
7 $c2w := P_{|P|} \rightarrowtail CI$
8 $P' :=$ Etendre P avec $c2w$
9 **si** $V_{dest} \supseteq \{$sommets origine de $occur_G(c2w)\}$ **alors**
`// P' ⊒ c2w, il n'y aura pas besoin d'étendre c2w`
10 Marquer $c2w$ dans $LC2W$
`// Nous ajoutons l'extension à l'arbre préfixé`
11 Ajouter un fils $pt_{child} := \{P', \sigma_{P'}(c2w)\}$ à pt
12 `EtendreChemin` (G,$LC2W$,P',pt_{child})

Exemple Considérons le premier a-dag de la figure 1 sur lequel nous appliquons l'algorithme avec $minsup = 3$. Nous prenons le chemin pondéré de taille 2 $cd \overset{3}{\rightarrowtail} bi$, dont les occurrences sont $3 \rightarrowtail 6$, $4 \rightarrowtail 7$ et $5 \rightarrowtail 7$, et nous essayons de l'étendre vers le bas (ses sommets destinations sont donc 6 et 7), comme montré dans la partie droite de la figure 3. Les items candidats pour l'extension (ceux appartenant à au moins un fils de chaque sommet destination) sont f, g, h et i mais pas e puisque le sommet 6 n'a aucun fils dont l'itemset associé contient e. Algo.2 lignes 4,5 nous donne l'itemset fermé h supporté par les arcs $6 \rightarrowtail 8$, $7 \rightarrowtail 8$ et $7 \rightarrowtail 9$ (mais pas l'itemset $fghi$ car son support est 2, c-à-d inférieur au $minsup$).

Nous pouvons maintenant ajouter l'itemset h au chemin comme le montre la figure 3, sur laquelle les occurrences, bien que non stockées dans les arbres préfixés, sont représentées en dessous de chaque chemin pondéré de taille 2. Les branches coupées indiquent que l'extension viole la contrainte de support. Comme l'extension $bi \overset{3}{\rightarrowtail} h$ couvre toutes les occurrences de $bi \rightarrowtail h$ dans le a-dag, ce chemin pondéré n'est pas condensé. Nous le marquons donc dans $LC2W$ (algo.2 lignes 9,10).

Les deux arbres préfixés illustrés dans la figure 3 représentent toutes les extensions vers le bas et le haut de $cd \overset{3}{\rightarrowtail} bi$. Finalement, nous générons les deux chemins pondérés condensés $ah \overset{3}{\rightarrowtail} cd \overset{3}{\rightarrowtail} bi \overset{3}{\rightarrowtail} h$ et $a \overset{5}{\rightarrowtail} cd \overset{3}{\rightarrowtail} bi \overset{3}{\rightarrowtail} h$.

5 Résultats expérimentaux

Nous avons implémenté l'algorithme en C++. Les expériences ont été réalisées sur un ordinateur avec Ubuntu 12.04 LTS et un Intel Core i5 @ 3.20GHz avec 8Go de mémoire RAM. Les expériences portent sur un jeu de données réel 'Dengue' et quatre jeux synthétiques. Le jeu réel a 223 sommets, 984 arcs, et on associe à chaque sommet entre 10 et 11 attributs. Les deux jeux synthétiques générés à partir de V20K-E60K ont 20000 sommets et 60000 arcs (avec 1 à 5 attributs et 5 à 10 attributs parmi 15), tandis que les deux jeux synthétiques générés à partir de V40K-E120K ont 40000 sommets et 120000 arcs (avec 1 à 5 attributs et 5 à 10 attributs parmi 15). Le nombre d'attributs est généré selon une loi normale centrée sur la taille moyenne voulue (respectivement 3 et 7.5).

La figure 4 compare les performances et le nombre de solutions entre l'extraction de la totalité des solutions et celle de leurs condensés seulement, pour le jeu de données Dengue. Nous pouvons apprécier le taux de compression, qui varie de ~ 10 jusqu'à $\sim 10^4$.

La figure 5 montre le passage à l'échelle de notre approche sur des données relativement grandes. Les graphiques du haut (où chaque sommet a entre 1 et 5 items parmi 15) montrent l'impact de la taille du a-dag sur le temps d'exécution. Les graphiques du bas (où chaque sommet a entre 5 et 10 items parmi 15) montrent que le nombre d'attributs impacte plus profondément les performances (le $minsup$ n'a pu être autant baissé, faute de mémoire suffisante).

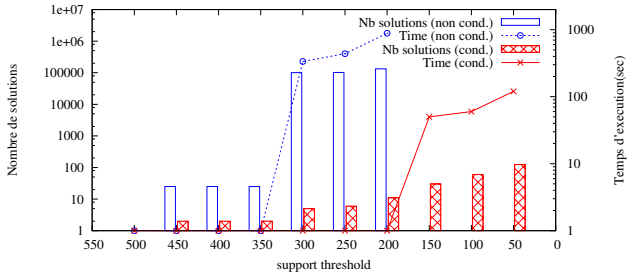


FIG. 4: *Dengue : Temps d'exécution et tailles des solutions (condensées et non condensées).*

6 Conclusion

Dans cet article, nous avons étudié l'extraction de motifs fréquents dans un unique DAG attribué. Nous avons introduit un nouveau type de motif et proposé une première représentation condensée exacte dans ce contexte. Nos expérimentations sur des jeux de données réelles et synthétiques soulignent le haut taux de compression de notre représentation condensée ainsi que l'efficacité de notre algorithme. Les travaux futurs pourraient porter sur l'étude de l'influence du choix de chemins pondérés de taille 2 sur les performances de l'algorithme. Une autre perspective serait d'étendre ces travaux à l'extraction de sous-graphes condensés dans le même contexte.

Remerciements. Ce travail a été financé par le contrat ANR-2012-COSI-012 FOSTER.

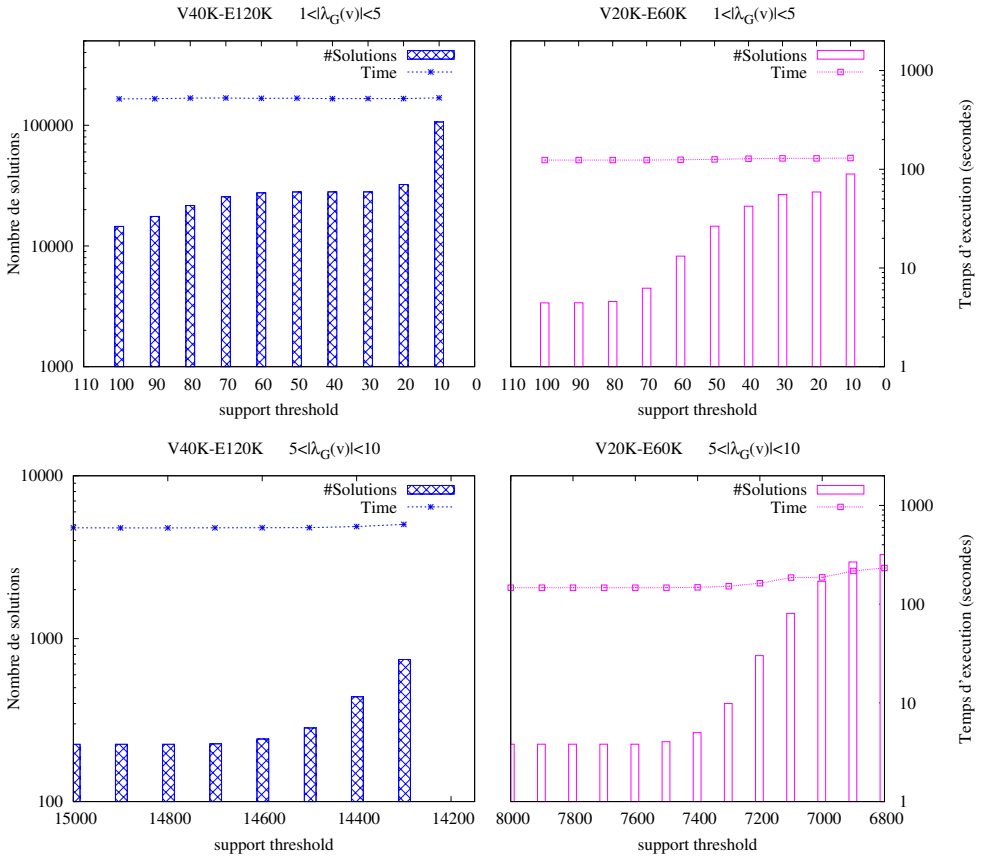


FIG. 5: *Temps d'exécution et tailles des solutions pour quatre jeux de données synthétiques.*

Références

Borgelt, C. et M. R. Berthold (2002). Mining molecular fragments : Finding relevant substructures of molecules. In *ICDM '02*, pp. 51–58.

Borges, J. et M. Levene (2000). A fine grained heuristic to capture web navigation patterns. *SIGKDD Explor. Newsl. 2*(1), 40–50.

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.-F. Boulicaut (2008). Data peeler : Contraint-based closed pattern mining in n-ary relations. In *SDM '08*, pp. 37–48.

Chen, M.-S., J. S. Park, et P. S. Yu (1998). Efficient data mining for path traversal patterns. *IEEE Trans. on Knowl. and Data Eng. 10*(2), 209–221.

Cook, D. J. et L. B. Holder (1994). Substructure discovery using minimum description length and background knowledge. *J. Artif. Int. Res. 1*(1), 231–255.

Fiedler, M. et C. Borgelt (2007). Support computation for mining frequent subgraphs in a single graph. In *Mining and Learning with Graphs, MLG '07*.

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.

Geng, R., W. Xu, et X. Dong (2007). Wtpminer : efficient mining of weighted frequent patterns based on graph traversals. In *KSEM '07*, pp. 412–424.

Gudes, E., S. E. Shimony, et N. Vanetik (2006). Discovering frequent graph patterns using disjoint paths. *IEEE Trans. Knowl. Data Eng. 18*(11), 1441–1456.

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.

Kuramochi, M. et G. Karypis (2005). Finding frequent patterns in a large sparse graph*. *Data Min. Knowl. Discov. 11*(3), 243–271.

Matsuda, T., T. Horiuchi, H. Motoda, et T. Washio (2000). Extension of graph-based induction for general graph structured data. In *PAKDD '00*, pp. 420–431.

Miyoshi, Y., T. Ozaki, et T. Ohkawa (2009). Frequent pattern discovery from a single graph with quantitative itemsets. In *ICDMW '09*, pp. 527–532.

Moser, F., R. Colak, A. Rafiey, et M. Ester (2009). Mining cohesive patterns from graphs with feature vectors. In *SDM '09*, pp. 593–604.

Nanopoulos, A. et Y. Manolopoulos (2001). Mining patterns from graph traversals. *Data Knowl. Eng. 37*(3), 243–266.

Pasquier, N., Y. Bastide, R. Taouil, et L. Lakhal (1999). Discovering frequent closed itemsets for association rules. In *ICDT '99*, pp. 398–416.

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 '07*.

Wang, J., W. Hsu, M. L. Lee, et C. Sheng (2006). A partition-based approach to graph mining. In *ICDE '06*, pp. 74–. IEEE Computer Society.

Washio, T. et H. Motoda (2003). State of the art of graph-based data mining. *SIGKDD Explor. Newsl. 5*(1), 59–68.

Yan, X. et J. Han (2003). Closegraph : mining closed frequent graph patterns. In *KDD '03*, pp. 286–295.

Yan, X., J. Han, et R. Afshar (2003). Clospan : Mining closed sequential patterns in large datasets. In *SDM '03*, pp. 166–177.

Summary

Directed acyclic graphs can be used accross many application domains. In this paper, we study a new pattern domain for supporting their analysis. Therefore, we propose the pattern language of weithed paths, primitive constraints that enable to specify their relevancy (e.g., frequency and compactness constraints), and algorithms that can compute the specified collections. It leads to a condensed representation setting whose efficiency and scalability are empirically studied.