Weighted Path as a Condensed Pattern in a Single Attributed DAG [*]

Jérémy Sanhes and **Frédéric Flouvat** and **Nazha Selmaoui-Folcher**
PPME, Université de Nouvelle Calédonie
BP R4, F-98851 Nouméa, New Caledonia

Claude Pasquier
Institute of Biology Valrose
CNRS UMR7277 - 06108 Nice, France

Jean-François Boulicaut
Université de Lyon, CNRS, INSA de Lyon
LIRIS UMR5205, 69621 Lyon, France

Abstract

Directed acyclic graphs can be used across many application domains. In this paper, we study a new pattern domain for supporting their analysis. Therefore, we propose the pattern language of weighted 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.

1 Introduction

Graphs are ubiquitous in many data analysis settings. Recently, richer graph models have been considered where, for instance, vertices or edges are labelled by sets of attributes or properties (i.e., itemsets) instead of a single one. For example, a social network can be represented as a large graph where each vertex denotes a person and its associated domains of interests. These graphs are called attributed graphs or itemset-associated graphs [Fukuzaki *et al.*, 2010]. Quite often, for instance when time is concerned, edges are oriented and graphs turn to be acyclic.

Among others, we are concerned with the analysis of spatio-temporal data that can be modeled by means of attributed directed acyclic graphs (a-DAG). In an a-DAG, vertices may denote spatial objects characterized by a set of attributes and/or events while edges may represent the spatio-temporal proximity between objects (i.e., neighbouring objects in consecutive timestamps). Our goal is to tackle such data analysis problems where studied objects are not stationary. Indeed, a specificity of geographical data is that "everything is related to everything else, but near things are more related than distant things" (Tobler's first law of geography). This statement reflects the concept of spatial dependence between geographical objects. In such a spatial analysis setting, a DAG is a natural representation to model spatial dependences between objects at different times (i.e., modeling the influence of one object on an other). For example, it can be used to model the spatial dependences between living areas when studying the diffusion of a virus, or to model dependences between erosion objects and their environments when looking for a better understanding of soil erosion dynamics. Indeed, in an a-DAG graph representing the spread of a vector-borne disease in a city (e.g., the Dengue fever), vertices could be the city districts at a given timestamp and they may be characterized by a set of attributes or events. Here, edges may express the disease propagation from one area to an other at successive timestamps. If we consider now soil erosion analysis, vertices may be geological objects like, e.g., gullies, that are observed at a given date. Their characteristics could be expressed by itemsets, and edges would be used to symbolize geological events like the merge or the division of the related objects (see Fig.1).

Usually, when tackling such application domains, well known diffusion models are assumed and exploited. We are looking for a generic data mining perspective that does not rely on any a-priori domain knowledge. We study frequent weighted path mining in a single a-DAG where each weight denotes the frequency of a transition. Frequent paths are useful to support the analysis of the causal relationship between sequences of events and/or attributes. Since the number of path patterns can be huge, we design a condensed representation of such collections.

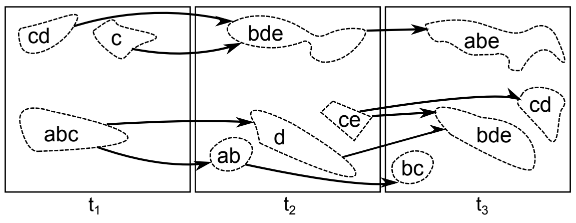


Figure 1: Example of an a-DAG built from successive timestamps containing several areas characterized by a set of attributes. An edge represents the evolution of an object to an other through two consecutive timestamps.

Our work is related to sequential data mining and graph mining. Many algorithms have been proposed to support fre-

[*]This work is supported by ANR-COSI-012 FOSTER french contract.

quent sequence mining. Closedness, which has been extensively studied for itemset mining [Pasquier *et al.*, 1999] has been applied to sequences as well [Yan *et al.*, 2003]. A pattern is closed if there is no super-patterns (w.r.t. a specialization/generalization relation) with the same support (i.e., occurring in the same transactions or sequences). Also, several algorithms have been proposed to mine patterns in graph transactions [Inokuchi *et al.*, 2000; Borgelt and Berthold, 2002; Washio and Motoda, 2003; Yan and Han, 2002]. Condensed representations of frequent graph patterns have been studied [Yan and Han, 2003; Termier *et al.*, 2007]. For example, [Yan and Han, 2003] proposes to mine closed frequent graphs in graph transactions, and [Termier *et al.*, 2007] adapts this approach for transactions of DAGs.

These studies focus on frequent pattern mining in transactions of graphs while we want to consider a single (large) DAG. The graph-transaction setting and the single-graph setting share common properties but the algorithms developed for the former cannot be used for the latter whereas the opposite is true [Kuramochi and Karypis, 2005]. One of the first problems in the single-graph setting is how to define pattern frequency. Indeed, it cannot be defined as the number of transactions in which it occurs. Several authors have studied this issue [Kuramochi and Karypis, 2005; Fiedler and Borgelt, 2007; Bringmann and Nijssen, 2008]. Most of them define a frequency based on pattern occurrences but this is not simple: several occurrences may overlap, leading to a non-monotonic frequency measure. However, exploiting such frequency measures, several algorithms have been proposed to mine frequent patterns in a single graph [Cook and Holder, 1994; Matsuda *et al.*, 2000; Kuramochi and Karypis, 2005; Gudes *et al.*, 2006]. When tractable, the number of frequent patterns can be huge and it makes sense to look for condensed representations of them, for instance closed ones. To the best of our knowledge, closedness has not yet been studied within the single-graph setting.

Moreover, most of graph mining algorithms process labeled graphs, i.e., graphs with only one label associated to each vertex and/or edge. Mining our attributed graphs leads to a combinatorial explosion (i.e., the exploration of search spaces for both graphs and label itemsets). Few works have considered attributed graphs. [Miyoshi *et al.*, 2009] mines a labeled attributed graph, i.e., a graph with labels and quantitative itemsets in vertices. By keeping labels, frequent pattern mining is simplified and decomposed in two steps: mining the labeled graph and mining the itemsets. [Moser *et al.*, 2009; Fukuzaki *et al.*, 2010] focus on mining cohesive patterns and itemset-sharing patterns, i.e., patterns representing subgraphs with shared itemsets.

Computing frequent paths has been already studied as well. For example, [Chen *et al.*, 1998] mines frequent path traversal patterns in a labeled directed acyclic graph representing user accesses in web pages. Traversal patterns have also been considered in [Borges and Levene, 2000; Nanopoulos and Manolopoulos, 2001; Geng *et al.*, 2007]. However, in these works, the authors only consider classical labeled graphs and not attributed graphs. Moreover, their definition of a path is different from the one we use: they can push more stringent constraints like avoiding repeated vertices and/or edges.

The same data mining problem has been already tackled by some of us in [Mabit *et al.*, 2011]. In this paper, the authors propose to flatten the graph and then to mine frequent sequences using available algorithms from the shelve. Unfortunately, their experiments show that this is not scalable due to the intrinsic complexity of this mining task. Here, we avoid a sequence mining approach for two reasons:

- A graph contains structural information that cannot be exploited if it is decomposed into sequences. This can motivate for a direct single graph mining approach.
- If we still want to decompose the graph into several sequences, we must choose to either duplicate nodes or delete edges. The last solution seems unacceptable since we would loose information. Furthermore, duplicating nodes will over-express information and thus add a bias. Although this duplication can be managed, it remains ineffective [Mabit *et al.*, 2011; Selmaoui-Folcher and Flouvat, 2011].

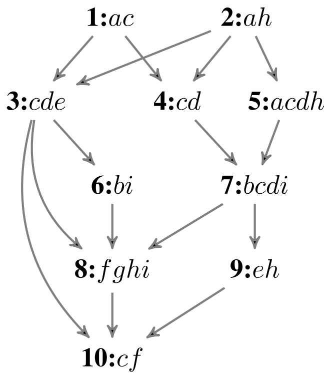
Our contribution is twofolds. First, we propose an original definition of what could be an (exact) condensed representation in a single-graph setting (see Section 3). Second, we propose an efficient algorithm to mine frequent patterns, i.e., weighted paths, in a single a-DAG (see Section 4). Our experiments highlight its good performances and the high compression rate provided by the designed condensed representation (see Section 5).

2 Preliminaries

Let us introduce some needed definitions or concepts about attributed directed acyclic graphs and weighted paths.

2.1 Attributed directed acyclic graphs

Attributed DAG. An attributed DAG (or **a-DAG**) $G = (V_G, E_G, \lambda_G)$ on a set of items $\mathcal{I}$ consists of a set of vertices V_G, a set of directed edges $E_G \subseteq V_G \times V_G$ and a labelling function $\lambda_G : V_G \to \mathcal{P}(\mathcal{I})$ that maps each vertex of the DAG G to a subset of $\mathcal{I}$ ($\mathcal{P}(\mathcal{I})$ denotes the power set of $\mathcal{I}$). An example of such a graph is given in Fig. 2.



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

Figure 2: An a-DAG example.

Path pattern and path occurrence. Let P be a succession of itemsets $I_i \in \mathcal{P}(\mathcal{I})$ denoted $P = I_1 \rightarrowtail I_2 \rightarrowtail \cdots \rightarrowtail I_{|P|}$. P is called a **path pattern** iff there exist a succession of vertices $v_1, v_2, \dots, v_{|P|} \in V_G$ satisfying $\forall I_i \in P$, $I_i \subseteq \lambda_G(v_i)$ and such that each v_i is a parent of v_{i+1} in G. The succession of vertices $O = v_1 \rightarrowtail v_2 \rightarrowtail \cdots \rightarrowtail v_{|P|}$ is called a **path occurrence** of P. For example, given the a-DAG in Fig. 2, the occurrences of the **size-3** path $ah \rightarrowtail cd \rightarrowtail i$ are $2 \rightarrowtail 3 \rightarrowtail 6$, $2 \rightarrowtail 3 \rightarrowtail 8$, $2 \rightarrowtail 4 \rightarrowtail 7$, $2 \rightarrowtail 5 \rightarrowtail 7$, and $5 \rightarrowtail 7 \rightarrowtail 8$. The occurrence $2 \rightarrowtail 3 \rightarrowtail 6$ **supports** the paths $ah \rightarrowtail cde \rightarrowtail bi$, $a \rightarrowtail cde \rightarrowtail bi$, $h \rightarrowtail cde \rightarrowtail bi$, $h \rightarrowtail c \rightarrowtail bi$, and so on.

Let P_i denote the i^{th} itemset of P, O_i be the i^{th} vertex of O, and $occur_G(P)$ be the set of P occurrences in G.

2.2 Weighted paths

Simple path patterns describe properly a sequence of events in an a-DAG. It would be however useful to know the contribution of every single edge to pattern occurrences. Therefore, we propose the pattern language of weighted paths.

Weighted path. Weighted paths are paths with a weight on each edge that represents the number of its different occurrences over all the path occurrences. For example, in the data from Fig. 2, the path $P = ah \rightarrowtail cd \rightarrowtail i$ whose occurrences have been listed earlier provides the pattern:

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

Indeed, the number of different occurrences of $ah \rightarrowtail cd$ in $occur_G(P)$ is 4 and the number of different occurrences of $cd \rightarrowtail i$ in $occur_G(P)$ is 5. Such a presentation supports pattern interpretation: it tells that itemset ah occurs 4 times before path $cd \rightarrowtail i$ occurs, or that itemset i occurs 5 times after path $ah \rightarrowtail cd$ occurs. From now on, $\omega_G(P_i \rightarrowtail P_{i+1})$ designates the weight of the edge between itemsets P_i and P_{i+1}.

Inclusion relation. We define the operator $\sqsubseteq$ on a couple of weighted paths as follows: $P \sqsubseteq P'$ iff $|P| \leq |P'|$ and $\exists k \in [0, |P'| - |P|]$ such that

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

Inclusion of itemsets and weights equality are checked both. A weighted path P' absorbs/captures another weighted path P if we can find P in a sub-sequence of P'. We say that a weighted path P' is a **super-weighted path** of P, or that P' **contains** P.

2.3 Problem statement

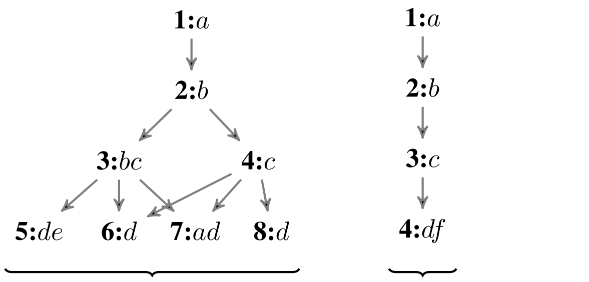
A popular data mining problem concerns frequent pattern discovery, i.e., looking for patterns with a support/frequency that is greater than a given threshold. Based on [Bringmann and Nijssen, 2008][1], we define an anti-monotonic support value of a pattern P in an a-DAG G denoted $\sigma_G(P)$:

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

[1]This measure can be either applied on vertices or edges

While being easily computable and anti-monotonic, this support fits well with the notion of frequency in a single DAG: it distinguishes paths occurring at different locations in a DAG from those finishing on or beginning from few edges. In other words, we want totally distinctive paths to be better valued than paths that share many edges.

Weighted paths actually help to better describe the evolution from an itemset to another (see Fig.3). Simple paths would have blurred such distinction between some patterns having the same support but occurring in varying ways.



a-DAG A
$occur_A(a \rightarrowtail b \rightarrowtail c \rightarrowtail d) = \{1 \rightarrowtail 2 \rightarrowtail 3 \rightarrowtail 5,\ 1 \rightarrowtail 2 \rightarrowtail 3 \rightarrowtail 6,\ \vdots\ 1 \rightarrowtail 2 \rightarrowtail 3 \rightarrowtail 8\}$

a-DAG B
$occur_B(a \rightarrowtail b \rightarrowtail c \rightarrowtail d) = \{1 \rightarrowtail 2 \rightarrowtail 3 \rightarrowtail 4\}$

Figure 3: Two a-DAGs where path $a \rightarrowtail b \rightarrowtail c \rightarrowtail d$ occurs in different ways but has the same support (1).

The collection of frequent paths in G is the set of patterns P such that $\sigma_G(P) \geq minsup$, $minsup$ being a user-defined threshold. However, the number of frequent paths in G can be huge. In such a situation, it makes sense to look at the concept of condensed representation of frequent patterns [Calders *et al.*, 2004].

Given an a-DAG G, we look for the condensed representation denoted $cond(G)$ of all weighted paths: each frequent weighted path *and its support* must be derivable from $cond(G)$.

3 Exact condensed representation of frequent paths

Many authors have been working on closed pattern mining (see, e.g., [Pasquier *et al.*, 1999; Yan *et al.*, 2003; Yan and Han, 2003]). In these settings, (frequent) closed patterns form an exact condensed representation of the frequent patterns: the computed collection of closed patterns is much smaller while it is possible to deduce the set of all frequent patterns from them.

Closed and condensed. Let us first highlight the differences between condensed representations from either a single graph or graph transactions. In the latest, the most popular form of condensed representation of frequent patterns is the collection of closed patterns. It exploits the Galois connexion that holds between transactions and patterns. An important property of the closure operator in this context is the support preservation property: patterns and their closures have the same support. This property relies on the support definition: a pattern is counted once per transaction. In the single-graph setting, the support definition is quite different: it is the minimum weight over the path edges. Moreover, the Galois connexion defined for closed patterns cannot be applied since we do not have transactions here. Another problem is that a pattern has only one associated closed pattern. However, a weighted path can be deduced from several different super-weighted-paths, as shown in Fig. 4. It turns out that we can hardly use a closure approach in our setting.

$$\begin{tikzcd}
\textbf{1:}ab \arrow[dr] & & \textbf{2:}b \arrow[dl] \\
& \textbf{3:}c \arrow[d] & \\
& \textbf{4:}d &
\end{tikzcd}$$

weighted path	condensed candidates
$c \overset{1}{\rightarrowtail} d$	$ab \overset{1}{\rightarrowtail} c \overset{1}{\rightarrowtail} d$ $b \overset{2}{\rightarrowtail} c \overset{1}{\rightarrowtail} d$

Figure 4: A weighted path can be included in several super-weighted paths.

A condensed representation of paths. As the support of a weighted pattern is directly encoded in it (i.e., the support is the minimum weight over all its edges), we can define the set of condensed weighted paths in an a-DAG G as follows:

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

Note that there is no need to check for support equality since the support information is already attached to the pattern itself.

Theorem 3.1. *Each path of G as well as its support can be deduced from $cond(G)$.*

Proof. *a*) From the definition of weighted paths and σ_G, the latter can be deduced from the former (minimum of weights). Reading a weighted path provides its support. *b*) Inclusion relation uses both information about attributes and edges weights; Then, a weighted path can be deduced from any of its super-weighted paths. □

4 Mining condensed weighted paths

We propose a two-steps algorithm to mine condensed weighted paths directly from the graph structure. Unlike [Mabit *et al.*, 2011] who first tries to flatten the graph, we avoid such an expensive and unnecessary candidate generation. It can handle the inherent high memory complexity of the problem (as shown later in Section 5). In addition, the structural information is kept that enables to push structural constraints if needed.

First, we mine every single size-2 weighted path (with one edge) that is condensed regarding the set of size-2 (only) weighted patterns. Then, we use a depth-first search to retrieve the condensed set by extending previously computed size-2 weighted paths.

4.1 Mining size-2 weighted paths

Size-2 weighted paths can naturally be represented by triplets like $\{\omega_G(P_{orig} \rightarrowtail P_{dest}), I_{orig}, I_{dest}\}$. Those triplets match frequent triadic concepts such as defined by [Cerf *et al.*, 2008]. We use their algorithm to mine size-2 weighted paths.

Proposition 4.1. *Given the set of size-2 weighted paths called $L2W$, and the following ternary relation:*

1. *The first dimension is $\mathcal{P}(E_G)$, the second and the third are $\mathcal{P}(\mathcal{I})$,*
2. *Given an edge $v_1 \rightarrowtail v_2 \in E_G$, the equivalent tuple $T \in (E_G, \mathcal{P}(\mathcal{I}), \mathcal{P}(\mathcal{I}))$ is $T = (v_1 \rightarrowtail v_2, \lambda_G(v_1), \lambda_G(v_2))$,*

*the closed 3-sets (i.e., size-2 paths) noted $LC2W$ **are equivalent to** condensed size-2 weighted paths w.r.t $L2W$.*

Proof. Let us take up the definition of closed 3-set from [Cerf *et al.*, 2008]: a 3-set S is closed iff there is no other 3-set S' such that $\forall i \in \{1, 2, 3\}, S^i \subseteq S'^i$. As Dimension 2 and Dimension 3 represent respectively origin and destination itemsets, the definition of closed patterns on $(E_G, \mathcal{I}, \mathcal{I})$ is the definition of condensed size-2 weighted paths (but only condensed w.r.t. *size-2* weighted paths). Their weight is the number of different edges, which is actually the size of S^1. □

For example, we use the closed 3-set $\{\{1 \rightarrowtail 3, 1 \rightarrowtail 4\}, \{b, c\}, \{c, d, e\}\}$ to provide the weighted path:

$$P = bc \overset{2}{\rightarrowtail} cde.$$

Having such sets of size-2 weighted paths enables to proceed to a standard depth-first search extension from any previously found size-2 weighted path.

4.2 Extending weighted paths

The goal of the second step is to extend weighted paths until they get condensed. To do so, we use the previously computed set of size-2 weighted paths which are ensured to be maximal w.r.t. itemsets on both origin and destination vertices. Their extension must be executed both downward (by adding children) and upward (by adding parents). A single a-DAG vertex can have several children and parents. To deal with the many possible combinations of extensions, we use two prefix trees (one for each extension direction), as shown in Fig.5. Due to space limitations, we only present downward extension. Upward extension is done following the same principle and properties (by simply considering parent vertices instead of child vertices).

Given a pattern P, we define $V_{dest}(P) = \{v_{|P|} \in occur_G(P)\}$ and $Li(P) = \{i \in \mathcal{I}$, such that $\forall v \in V_{dest}(P), v$ has at least one child u s.t. $i \in \lambda_G(u)\}$. $Li(P)$ is the list of items belonging to at least a child of each destination vertex of P. The extension method is based on the following proposition that is derived from Proposition 4.1.

Proposition 4.2. *Let P be a weighted path to be extended (downward). Let the projected binary relation of P called $DB|_P$ be the following:*

1. *The first dimension is $\mathcal{P}(E_G)$, the second is $\mathcal{P}(Li(P))$,*

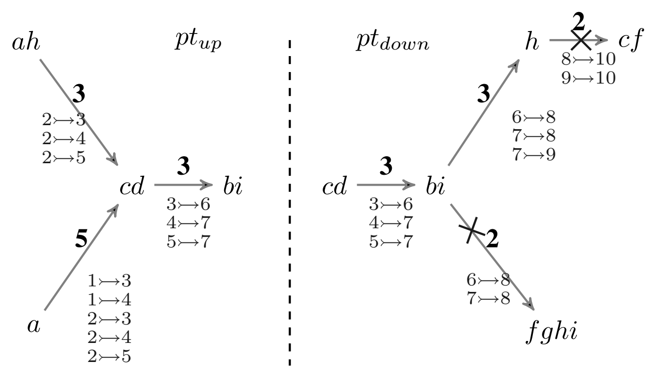


Figure 5: Prefix trees for extensions of $cd \overset{3}{\rightarrowtail} bi$ (see Fig. 2, $minsup = 3$)

2. *Given an edge $v_1 \rightarrowtail v_2 \in E_G$ such that $v_1 \in V_{dest}(P)$, the corresponding couple $T \in (E_G, \mathcal{P}(Li(P)))$ is $T = (v_1 \rightarrowtail v_2, \lambda_G(v_2) \cap Li(P))$.*

The extended weighted paths $\{P \rightarrowtail u \mid u$ is closed in $DB|_P\}$ are super-weighted paths of P. This way, their condensed forms are simply obtained by extending them recursively.

Proof. First, the foregoing extension guarantees both itemsets inclusion and weights preservation. Following the extension, we obtain a super-weighted path of P. If this one cannot be extended anymore, then it is a condensed one. Indeed, a super-pattern having the same weights cannot exist (because of the definition of closedness in $DB|_P$, each itemset being closed and obtained by successive projections of the base). The last recursively generated super-weighted paths are then condensed. □

Algorithm. Each size-2 weighted path previously found (that might be not condensed globally) is recursively extended until it becomes condensed (Algo. 1 Lines 3-8)[2]. To extend a path P, we look at $V_{dest}(P)$, the destination vertices of $occur_G(P)$ (the last vertices of $occur_G(P)$). If path P can be extended with itemset I, then each destination vertex of $occur_G(P)$ must have at least one child whose associated itemset includes I. This set of itemsets obtained from $occur_G(P)$ constitutes the projected database of P ($DB|_P$, Algo. 2 Line 2).

From this projection, we extend the path with itemsets satisfying the condensed representation definition. To that end, we mine closed itemsets in the projected database $DB|_P$ (Algo. 2 Line 5) as explained in Proposition 4.2. Extension is then performed (Algo. 2 Lines 6-12) for every generated itemset that satisfies the support constraint. If one of the extended patterns captures a size-2 weighted path (Algo. 2 Line 9), it will not be proposed for extension in the next iterations (Algo. 1 Lines 2,3).

[2]For readability purposes, the exponent $\cdot^{-1}$ applied to a set of edges means that we reverse edge directions for each edge of this set. It is only used to define an extension towards parents (upward extension) instead of children.

Algorithm 1: Process each size-2 weighted path.

```
Input : $LC2W$, a-DAG $G$
1 while $LC2W \neq \emptyset$ do
2     Pick and remove $c2w$ from $LC2W$
3     Create two prefix trees $pt_{down}$ and $pt_{up}$
      // Extend downwards and upwards
4     ExtendPath ($G$,$LC2W$,$c2w$,$pt_{down}$)
5     ExtendPath ($G^{-1}$,$LC2W^{-1}$,$c2w^{-1}$,$pt_{up}$)
6     Generate solutions from $pt_{down}$ and $pt_{up}$
```

Algorithm 2: Extend path.

```
Input : $LC2W$, a-DAG $G$, weighted path $P$ applying
        for extension and $pt$ the prefix tree
Output: $LC2W$, $pt$
1 $V_{dest}(P) := \{$destination vertices of $occur_G(P)\}$
2 if $\exists v \in V_{dest}(P)$ such that $v$ has no child then stop
3 $Li(P) := \{i \in \mathcal{I}$, such that $\forall v \in V_{dest}(P)$, $v$ has at least
  one child $u$ such that $i \in \lambda_G(u)\}$
  // $Li(P)$ is the list of items that are
     at least in one child of each
     destination vertex of $P$
4 $DB|_P := \{$transactions $T = \{v \rightarrowtail u,\ i_1 i_2 \dots i_N\}$ s.t.
  $i_k \in Li(P) \cap \lambda_G(u)\}$
5 $LCI := \{$closed itemsets $CI$ of $DB|_P$ such that
  $\sigma(CI) \geq minsup\}$
  // Support of each $CI$ is the number
     of edges
6 foreach $CI \in LCI$ do
7     $c2w := P_{|P|} \rightarrowtail CI$
8     $P' :=$ Extend $P$ with $c2w$
9     if $V_{dest}(P) \supseteq \{$origin vertices of $occur_G(c2w)\}$
      then
          // $P' \sqsupseteq c2w$, no need to extend $c2w$
10        Remove $c2w$ from $LC2W$
      // We append the extension to the
         prefix tree
11    Append child $pt_{child} := \{P', \sigma_{P'}(c2w)\}$ to $pt$
12    ExtendPath ($G$,$LC2W$,$P'$,$pt_{child}$)
```

Example. Consider the first a-DAG in Fig. 2 on which we apply the algorithm with $minsup = 3$.

We take the size-2 weighted path $cd \overset{3}{\rightarrowtail} bi$, whose occurrences are 3↣6, 4↣7, and 5↣7, and we try to extend it downwards (its destination vertices are then 6 and 7), as shown in the right part of Fig. 5.

Candidate items for extension (those that belong to at least one child of each destination vertex) are f, g, h, and i but not e since Vertex 6 has no child containing e.

Algo. 2 Lines 4,5 provides the closed itemset h supported by the edges 6↣8, 7↣8, and 7↣9 (but not itemset $fghi$ because its support is 2, and thus lower than $minsup$).

We can now add itemset h to the path as shown in Fig. 5, on which occurrences (not stored in the prefix trees) are represented below each size-2 weighted path. Crossed branches indicate that the extension violates the support constraint.

As the extension $bi \overset{3}{\rightarrowtail} h$ actually covers all occurrences of $bi{\rightarrowtail}h$ in the a-DAG, this weighted path is not condensed. It is thus removed from $LC2W$ (Algo. 2 Lines 9,10).

The two prefix trees given in Fig. 5 represent all the upward and downward extensions of

$$cd \overset{3}{\rightarrowtail} bi.$$

At the end, it generates the two condensed paths:

$$ah \overset{3}{\rightarrowtail} cd \overset{3}{\rightarrowtail} bi \overset{3}{\rightarrowtail} h, a \overset{5}{\rightarrowtail} cd \overset{3}{\rightarrowtail} bi \overset{3}{\rightarrowtail} h.$$

5 Experiments

We implemented the algorithm in C++. Experiments were performed on a computer running Ubuntu 12.04 LTS and based on a Intel Core i5 @ 3.20GHz with 8GB main memory. Experiments have been run on a real 'Dengue disease' dataset and four synthetic datasets. The real dataset[3] has 223 vertices, 984 edges, and each vertex has 10 or 11 attributes. The two synthetic datasets generated from V20K-E60K have 20000 vertices and 60000 edges (with 1-5 items and 5-10 items among 15) while the two datasets generated from V40K-E120K have 40000 vertices and 120000 edges (with 1-5 items and 5-10 items among 15). The number of attributes follows a normal distribution whose mean is the desired size (respectively 3 and 7.5).

Fig. 6 shows a performance comparison between mining the whole set of solutions and mining only its corresponding condensed set for the Dengue dataset. The baseline method consists in applying the same search strategy as in Algo. 2 without trying to choose closed itemsets in projected databases (every candidate itemset has a chance to extend the current weighted path). One can appreciate the compression rate, which is $\sim$10 to up $\sim 10^4$ for the largest solution sets.

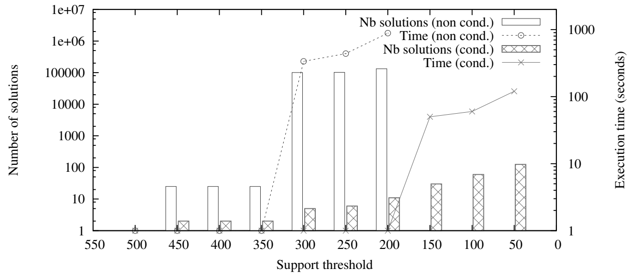


Figure 6: Run times and solution sizes (condensed vs. non condensed) for the 'Dengue' dataset

Fig. 7 shows the scalability of our approach on relatively large synthetic datasets. Top graphs (where each vertex has between 1 and 5 items among 15) illustrate the impact of the a-DAG size on the execution time. The bottom graphs (where each vertex has between 5 and 10 items among 15) show that itemsets size has a deeper impact on performance ($minsup$ could not be as much lowered due to a lack of memory).

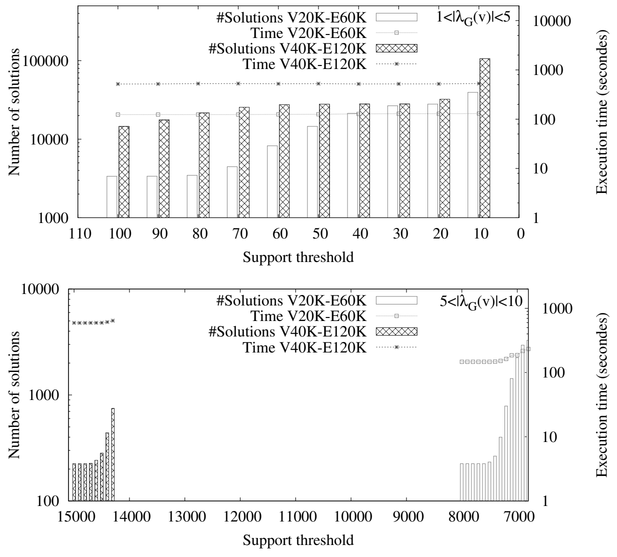


Figure 7: Time and size of condensed sets for synthetic datasets.

6 Conclusion

In this paper, we studied frequent pattern mining in a single attributed DAG. We introduced a new type of pattern and we proposed for the first time an exact condensed representation in the single-graph setting. Our experiments on real and synthetic datasets highlight the high compression rate of our condensed representation and the efficiency of our algorithm. A future work could be to study the influence of size-2 weighted paths on algorithm performances. Another perspective would be to extend this work to propose a condensed representation for frequent subgraph mining in a single-graph setting.

References

[Borgelt and Berthold, 2002] Christian Borgelt and Michael R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *ICDM '02*, pages 51–58, 2002.

[Borges and Levene, 2000] José Borges and Mark Levene. A fine grained heuristic to capture web navigation patterns. *SIGKDD Explor. Newsl.*, 2(1):40–50, 2000.

[Bringmann and Nijssen, 2008] Björn Bringmann and Siegfried Nijssen. What is frequent in a single graph? In *PAKDD '08*, pages 858–863, 2008.

[Calders *et al.*, 2004] Toon Calders, Christophe Rigotti, and Jean-François Boulicaut. A survey on condensed representations for frequent sets. In *Constraint-Based Mining and Inductive Databases*, pages 64–80, 2004.

[3]We thank the consortium "Prevention and prediction of dengue epidemics in New Caledonia" IRD-DASSNC-UNC-IPNC-MeteoFrance who gave us this dataset.

[Cerf *et al.*, 2008] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Data peeler: Contraint-based closed pattern mining in n-ary relations. In *SDM '08*, pages 37–48, 2008.

[Chen *et al.*, 1998] Ming-Syan Chen, Jong Soo Park, and Philip S. Yu. Efficient data mining for path traversal patterns. *IEEE Trans. on Knowl. and Data Eng.*, 10(2):209–221, 1998.

[Cook and Holder, 1994] Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. *J. Artif. Int. Res.*, 1(1):231–255, 1994.

[Fiedler and Borgelt, 2007] Mathias Fiedler and Christian Borgelt. Support computation for mining frequent subgraphs in a single graph. In *Mining and Learning with Graphs, MLG '07*, 2007.

[Fukuzaki *et al.*, 2010] Mutsumi Fukuzaki, Mio Seki, Hisashi Kashima, and Jun Sese. Finding itemset-sharing patterns in a large itemset-associated graph. In *PAKDD '10*, pages 147–159, 2010.

[Geng *et al.*, 2007] Runian Geng, Wenbo Xu, and Xiangjun Dong. Wtpminer: efficient mining of weighted frequent patterns based on graph traversals. In *KSEM '07*, pages 412–424, 2007.

[Gudes *et al.*, 2006] Ehud Gudes, Solomon Eyal Shimony, and Natalia Vanetik. Discovering frequent graph patterns using disjoint paths. *IEEE Trans. Knowl. Data Eng.*, 18(11):1441–1456, 2006.

[Inokuchi *et al.*, 2000] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD '00*, pages 13–23, 2000.

[Kuramochi and Karypis, 2005] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph*. *Data Min. Knowl. Discov.*, 11(3):243–271, 2005.

[Mabit *et al.*, 2011] Loïc Mabit, Nazha Selmaoui-Folcher, and Frédéric Flouvat. Modélisation de la dynamique de phénomènes spatio-temporels par des séquences de motifs. In *EGC*, pages 455–466, 2011.

[Matsuda *et al.*, 2000] Takashi Matsuda, Tadashi Horiuchi, Hiroshi Motoda, and Takashi Washio. Extension of graph-based induction for general graph structured data. In *PAKDD '00*, pages 420–431, 2000.

[Miyoshi *et al.*, 2009] Yuuki Miyoshi, Tomonobu Ozaki, and Takenao Ohkawa. Frequent pattern discovery from a single graph with quantitative itemsets. In *ICDMW '09*, pages 527–532, 2009.

[Moser *et al.*, 2009] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM '09*, pages 593–604, 2009.

[Nanopoulos and Manolopoulos, 2001] Alexandros Nanopoulos and Yannis Manolopoulos. Mining patterns from graph traversals. *Data Knowl. Eng.*, 37(3):243–266, 2001.

[Pasquier *et al.*, 1999] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT '99*, pages 398–416, 1999.

[Selmaoui-Folcher and Flouvat, 2011] Nazha Selmaoui-Folcher and Frédéric Flouvat. How to use "classical" tree mining algorithms to find complex spatio-temporal patterns? In *DEXA (2)*, pages 107–117, 2011.

[Termier *et al.*, 2007] Alexandre Termier, Yoshinori Tamada, Kazuyuki Numata, Seiya Imoto, Takashi Washio, and Tomoyuki Higuchi. Digdag, a first algorithm to mine closed frequent embedded sub-dags. In *MLG '07*, 2007.

[Washio and Motoda, 2003] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

[Yan and Han, 2002] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *ICDM '02*, pages 721–724, 2002.

[Yan and Han, 2003] Xifeng Yan and Jiawei Han. Closegraph: mining closed frequent graph patterns. In *KDD '03*, pages 286–295, 2003.

[Yan *et al.*, 2003] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In *SDM '03*, pages 166–177, 2003.