Data Min Knowl Disc (2006) 13:243–260
DOI 10.1007/s10618-006-0044-8

Support measures for graph data*

N. Vanetik · S. E. Shimony · E. Gudes

Received: 18 July 2005 / Accepted: 07 March 2006 / Published online: 26 May 2006

Abstract The concept of support is central to data mining. While the definition of support in transaction databases is intuitive and simple, that is not the case in graph datasets and databases. Most mining algorithms require the support of a pattern to be no greater than that of its subpatterns, a property called *anti-monotonicity*, or admissibility. This paper examines the requirements for admissibility of a support measure. Support measures for mining graphs are usually based on the notion of an instance graph—a graph representing all the instances of the pattern in a database and their intersection properties. Necessary and sufficient conditions for support measure admissibility, based on operations on instance graphs, are developed and proved. The sufficient conditions are used to prove admissibility of one support measure—the size of the independent set in the instance graph. Conversely, the necessary conditions are used to quickly show that some other support measures, such as weighted count of instances, are not admissible.

Keywords Data mining · Graph mining · Support measures

1. Introduction

The primary goal of datamining is to discover interesting patterns in data. Since patterns that appear frequently may be interesting or important, a primitive sub-task in achieving this goal is answering the question: how frequently does pattern **P** appear in a dataset *D*?

*Partially supported by the KITE consortium under contract to the Israeli Ministry of Trade and Industry, and by the Paul Ivanier Center for Robotics and Production Management.

N. Vanetik (✉) · S. E. Shimony · E. Gudes
Department of Computer Science, Ben-Gurion University of the Negev,
84105 Beer-Sheva, Israel
e-mail: orlovn@cs.bgu.ac.il

S. E. Shimony
e-mail: shimony@cs.bgu.ac.il

E. Gudes
e-mail: ehud@cs.bgu.ac.il

(We use the term *dataset* throughout the paper, to cover all sorts of data, including data in relational databases, semi-structured databases, web data, etc.) A similar question is: does the pattern **P** appear frequently in dataset D? The answer to these questions is then used to decide whether a pattern is interesting, either individually or in relation to other patterns. Usually the frequency of a pattern **P** is called the *support* of **P** (in D). An early appearance of this was in the classical paper by Agrawal et al. on mining association rules Agrawal and Srikant (1994).

Thus, a count of the number of pattern appearances in the dataset is a method commonly used to define the support measure. In the simplest case, a pattern is a set of items, and the dataset D is a set of transactions. The standard measure of support for itemsets in the literature is as follows. Let D be a set of transactions, and $I =< i_1, \ldots, i_k >$ be an item set. The support S of the itemset in the dataset is defined as $S(I) = \frac{|\{t|t\in D, <i_1,\ldots,i_k>\in t\}|}{|D|}$.

Usually in datamining tasks, a number $0 \leq \sigma \leq 1$, called the *minimum support* threshold is provided to the system. An itemset I that has $S(I) \geq \sigma$ is called *frequent*.

Defined in this manner, *the support of an item set I is always not greater than the support of any of the subsets of I*. This ***fundamental property*** of the support measure is important, because it is intuitively appealing, but also because of the following obvious corollary: an itemset I can be frequent, only if all of the subsets of I are frequent. The latter property (alternately called the *Apriori* principle, anti-monotonicity, or downward closure, in varied related work (Ng et al., 1998) has been of major importance in numerous datamining algorithms, used to prune candidate patterns and greatly improve performance. This property is the de-facto standard assumption for algorithms, which rely heavily on anti-monotonicity. These algorithms range from the a-priori algorithm for structured data (Agrawal and Srikant, 1994) to algorithms for mining paths (Chen et al., 1998), trees (Wang and Liu, 1998) and graphs (Kuramochi and Karypis, 2001).

Since data sets in many applications are not limited to transaction databases, and patterns are not limited to itemsets, a more general notion of support is required. A simple generalization of the scheme used in transaction databases is the following: The *support* of a pattern **P** in a dataset D is a measure of frequency of the instances of **P** in D. A *support measure* is a function $S : D \times \mathbf{P} \rightarrow \mathbb{R}_+ \cup \{0\}$ that for each pattern **P** in dataset D provides its support, a non-negative real number. A pattern **P** is called *frequent* if its support measure $S(\mathbf{P})$ is greater than or equal to a support threshold TS (for conciseness, we drop reference to the dataset D when unambiguously understood). Definitions of support in many types of datasets also usually observe the above fundamental property of support measures. The importance of the fundamental property of support measures (a.k.a. anti-monotonicity or downward closure) for itemsets indicates its general applicability, leading to the following definition:

Definition 1. (admissible support measure). A support measure S is *admissible* if for every dataset D and pattern **P** we have$S(\mathbf{P}) \geq 0$, and for every pattern $\mathbf{P}'$ such that $\mathbf{P}' \subseteq \mathbf{P}$ (meaning that $\mathbf{P}'$ is a subpattern of **P**) we have $S(\mathbf{P}') \geq S(\mathbf{P})$.

Whereas in the past, data mining was mainly applied to structured data and flat files, there is growing interest in mining and discovering frequent patterns in semi-structured data such as web data (Huffman and Baudin, 1997; Wang and Liu, 1998; Chamberlin, 2003), chemical compounds data (Wang et al., 2002) or biological data (Pennec and Ayache 1998). Although for itemsets and transaction databases, the obvious definition of support is admissible, it is not obvious that this is the case for other types of patterns.

Specifically, realizing that in data on the world-wide-web and in object databases, topology is meaningful, a datamining task that has been of increasing interest in the community is to

find frequent patterns (subgraphs) in a dataset (a large graph). In this case, there are several different intuitive ways to define support, but not all of them are admissible, as is shown in Section 2. In this paper, our goal is analysis of formal properties of the support measure, rather than graph mining algorithms. After formally defining the notion of an instance of a pattern in the dataset, we find necessary andsufficient conditions for admissibility (in the sense of the above definition) of a support measure. Our results are applicable to both directed and undirected graphs, and to both labeled and unlabeled graphs.

Finding an admissible support measure for graphs is not so easy. The naive support measure which counts the number of instances of a pattern in a graph is shown in Section 2 to be non-admissible. An intuitive support measure—size of maximum independent set of the instance graph (MIS)—is proposed and shown to be admissible. (see Section 2 for definition of instance graphs and MIS). The use of MIS as a support measure was first suggested in Vanetik et al. (2002), and was shown to be useful as a major component of an apriori-based algorithm for graph mining. It was later used also by Vanetik and Gudes (2004) and Kuramochi and Karypis (2001). Other support measures defined in the literature are examined in light of our results, as well as the generalization of our result beyond the graph mining domain. The major contribution of the present paper over (Vanetik et al., 2002) is the formal definition and the proofs for sufficient and necessary conditions for any admissible support measure.

The rest of the paper is organized as follows. Section 2 defines the notions of a graph pattern instances and instance graphs, examines the intuitive definitions of support for graphs, and defines the useful admissible MIS support measure. Section 3 defines operations on instance graphs, and uses them to show our main result, the necessary and sufficient conditions for admissibility. Section 4 shows that that the MIS measure is admissible, examines other support schemes, and discusses generalizations of our results. Finally, the related work is examined and compared to ours.

2. Instance graphs and basic support measures

We begin with assumptions about patterns and datasets, used in most of this paper. Henceforth, a *pattern* is assumed to be a labeled graph, either directed or undirected. A dataset is another (usually much larger) graph of the same type as the pattern. Although we assume in our derivation that labels, if they exist, are attached to nodes, all of our results apply for edge-labeled graphs as well. Labels are from some finite alphabet Σ. We also allow (and use in our proofs) unlabeled graphs, as these are equivalent to a special case where $|\Sigma| = 1$.

2.1. Instances and instance graphs

Let D be a (not necessarily connected) labeled dataset graph, and $\mathbf{P}$ be the pattern for which we are searching—a connected labeled graph.

Definition 2. Subgraph G of D is an *instance* of $\mathbf{P}$ in D just when there exists a label-preserving isomorphism between $\mathbf{P}$ and G.

Note that according to definition 2, instances of $\mathbf{P}$ in D are allowed to overlap, but two instances that have exactly the same set of edges and same set of vertices are considered to be the same instance. Let M be the set of all instances of $\mathbf{P}$ in D (i.e. subgraphs of D that are label isomorphic to G). Two instances $G_1, G_2 \in M$ are said to be *intersecting*, denoted

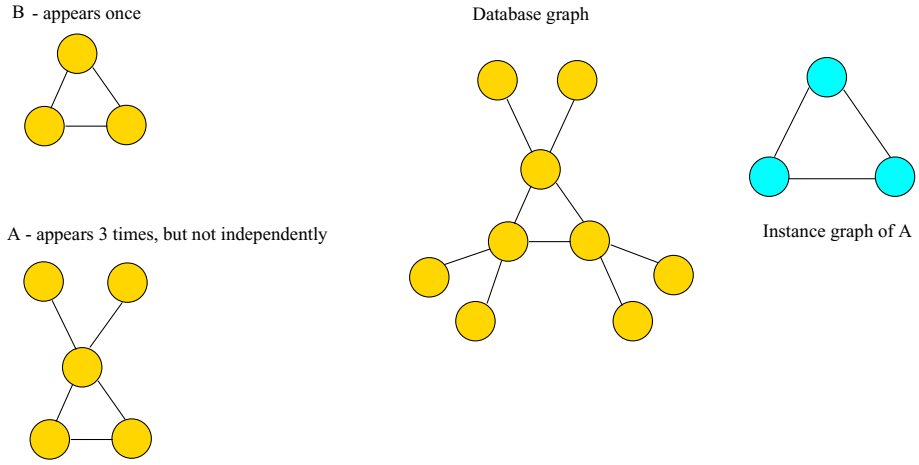


Fig. 1 Graph pattern support

by $G_1 \wedge G_2$, if they have at least one edge in common. If the instances do not have an edge intersection in D, this is denoted by $G_1 \|_D G_2$.

Definition 3. (Instance Graph). For a pattern $\mathbf{P}$ its *instance graph in* D, *denoted* $I_{\mathbf{P}}(D)$, is an undirected graph $(V_{\mathbf{P}}, E_{\mathbf{P}})$, with exactly one node standing for each instance of $\mathbf{P}$ in D. Denote by μ the mapping from nodes in $V_{\mathbf{P}}$ to instances (subgraphs of D). The edges in the instance graph are $E_{\mathbf{P}} = \{\{u, v\} \mid \mu(u) \wedge \mu(v)\}$. That is, an edge in the instance graph exists just when the respective instances intersect. When the dataset D is unambiguously understood, we sometimes omit D for brevity and use $I_{\mathbf{P}}$ to denote $I_{\mathbf{P}}(D)$. See Fig. 1 for an example, where the three instances of graph A mutually intersect in the database—hence the instance graph for A consists of a clique of size 3. Obviously, the instance graph for pattern B consists of a single node.

Definition 4. (Subpattern). Let D be a labeled graph, $\mathbf{P}$ a graph pattern, and p a subgraph of $\mathbf{P}$, denoted by $p \subset \mathbf{P}$. We call p a *subpattern* of $\mathbf{P}$. Likewise, we refer to $\mathbf{P}$ as a *superpattern* of p.

In the rest of the paper, $\mathbf{P}$ and p denote respectively a pattern and its subpattern in D. By definition, there exists a mapping μ from nodes of $I_{\mathbf{P}}$ to subgraphs of D, and from nodes of I_p to subgraphs of D. Let $u \in I_{\mathbf{P}}$ and $v \in I_p$. Whenever D and μ are implicitly understood, we denote the subgraph relation $\mu(v) \subset \mu(u)$ between instances in D, by $v \subset u$. For convenience, Table 1 summarizes our notation, as defined above and elsewhere in the paper.

2.2. Intuitive support measures for graphs

When considering graph patterns, we interpret set inclusion (as used in Definition 1) as the subgraph relationship. Therefore a support measure for graph patterns is admissible if the support for a graph patten $\mathbf{P}$ is never strictly greater than the support of any of its subgraphs.

Table 1 Notation used in this paper

D	Dataset
$\mathbf{P}, p$	Pattern and subpattern, respectively
$G_1 \wedge G_2$	Graphs G_1, G_2 have a common edge
$G_1 \| \| G_2$	Graphs G_1, G_2 have no common edge
$I_{\mathbf{P}}, I_p$	Instance graphs of $\mathbf{P}$ and p, respectively
μ	Mapping from nodes of an instance to subgraphs of D
$f(I_{\mathbf{P}})$	Induced support function
SubInstances(u)	Subgraphs of $\mu(u)$ isomorphic to p
SuperInstances(u)	Supergraphs of $\mu(u)$ isomorphic to $\mathbf{P}$
$I_{p\mathbf{P}}$	$I_{\mathbf{P}}$ modifed by series of clique contractions, additions, and edge removals
λ	Partial mapping from nodes of $I_{p\mathbf{P}}$ to the nodes of I_p.
π	Induced mapping from nodes of $I_{p\mathbf{P}}$ into the instances of p in D
δ	Maximum degree of $I_{\mathbf{P}}$ nodes
MIS	Maximum independent set measure

The most intuitive support measure is the number of instances of the pattern in a dataset. This measure, however, is not admissible. Figure 1 shows a dataset that contains 3 instances of pattern A and only one instance of pattern B, while $B \subseteq A$.

Another intuitive support measure that turns out not to be admissible is to count the number of possible isomorphisms between the pattern $\mathbf{P}$ and the subgraphs of D. This is equivalent to counting the instances, and multiplying the result by the number of automorphisms of $\mathbf{P}$. Again, the database in Fig. 1 is a counterexample, since $|Aut(B)| = 6$ and $|Aut(A) = 4|$ (where $Aut(\mathbf{P})$ is the set of automorphisms of graph $\mathbf{P}$). Thus, the support of A is 12, which is greater than 6, the support of B.

Clearly, simple admissible support measures exist—for example, any positive constant function is admissible. However, such a trivial support measure is less than useful for graph mining.

A more useful support measure is the MIS measure, defined as the size of the largest independent set of nodes in the instance graph.

(Recall that a set of nodes V is called independent if no pair of nodes in$V$ is connected by an edge). For example, in Fig. 1 the size of the independent set for both patterns is 1, maintaining admissiblity for this example. However, proving that a measure is admissible in general is much more difficult. Using the conditions for admissibility defined and proved next, we will show in Section 4 how one can easily prove whether a support measure (including MIS) is admissible or not.

3. Conditions for admissibility

Our conditions for admissibility are based on the instance graphs. Essentially, our main result (formally stated later on) is paraphrased as follows: an (instance-graph based) support measure is admissible if and only if it is non-decreasing under certain operations on the instance graph (clique contraction, node addition, and edge removal, defined below).

The motivation for these operations is that it is often easier to show that a support measure fulfills the conditions of the theorem, than to show admissibility of a measure by definition of admissibility. We begin by defining the operations used in our conditions for admissibility.

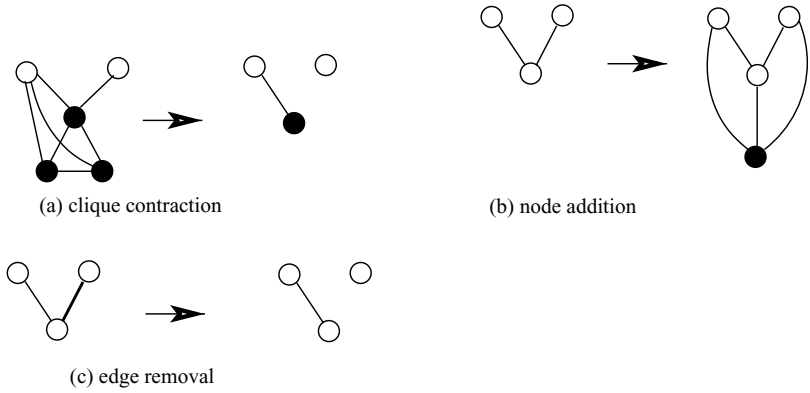


Fig. 2 Examples of operations on instance graphs

3.1. Operations on instance graphs

We define the following three operations on graphs: clique contraction (clique is defined as fully connected subgraph), node addition, and edge removal. Refer to Fig. 2 for examples of these operations. All the operations are assumed to be on an undirected, unlabeled, graph $G = (V, E)$, as we only use them on instance graphs.

Definition 5. (Clique Contraction). *Clique contraction* operates on a subgraph of G forming a clique $K = (V_K, E_K)$ in G, resulting in a graph $G' = (V', E')$ as follows. Let k be an arbitrary node in V_K representing the contracted clique. *The graph G' resulting from clique contraction* has the nodes $V' = V \backslash \mathrm{V_K} \cup \{\mathrm{k}\}$, and the edges:

$$E' = (E \cap \{\{u, v\} \mid u, v \in V'\}) \cup \{\{v, k\} \mid v \in V' - \{k\} \wedge \forall u \in V_K \ \{v, u\} \in E\} \qquad (1)$$

Intuitively, clique K is contracted into a single node k, which remains adjacent only to those vertices of V that were adjacent to every node of K in G. This definition applies to any clique, not necessarily a maximum clique. For the sake of uniformity, a subgraph consisting of a single node is considered to be a clique, and in this case clique contraction does not change the graph.

Definition 6. (Node Addition). *Node addition* operates on G and a new node u that is not in G, resulting in a new graph $G' = (V', E')$ whose node and edge sets are $V' = V \cup \{u\}$, $u \notin V$ and $E' = E \cup \{\{u, v\} \mid v \in V\}$.

Intuitively, node addition adds a new node to the graph, that has an edge with all other nodes in the graph.

Definition 7. (Edge Removal). *Edge removal* removes an edge e from G. The resulting graph is $G' = (V, E \backslash \{e\})$.

3.2. Sufficient conditions for admissibility

We formulate a sufficient condition for support measure admissibility in terms of operations on the instance graph. We restrict the discussion to support functions defined on the instance graph topology and combinatorics, as in the following remark (that is, the function can take into account anything that is based on node counts, edges between nodes, etc. but is not allowed to treat two nodes differently based on any node labels or identity, such as the identity of the instances the nodes represent in the dataset).

Remark. For every dataset D and pattern $\mathbf{P}$, a support function $f(\mathbf{P}, D)$ evaluates to a number in R^+. Whenever, for every pair $(\mathbf{P}, D)$, the instance graph $I_\mathbf{P}$ of $\mathbf{P}$ in D is defined unambiguously (as we have done for graph patterns and datasets), we define the induced support function $f(I_\mathbf{P}) = f(\mathbf{P}, D)$. We henceforth refer to the induced function, which we denote by $f(I_\mathbf{P})$, instead of referring to $f(\mathbf{P}, D)$.

Theorem 1. *A positive valued function $f(I_\mathbf{P})$ is an admissible support measure over graph patterns **if** it is non-decreasing under all of the following operations on $I_\mathbf{P}$:*

(A1) Clique contraction: $k = \text{Contract}(K)$, where K is a clique, and k is the node representing the clique after the contraction operator.
(A2) Node addition: $v = \text{Add}()$, where v is the new node added by the operator.
(A3) Edge removal: $\text{Remove}(e)$, where e is an edge to be removed.

Proof (outline): We prove that I_p can be obtained from $I_\mathbf{P}$ by applying only the operations **A1, A2, A3**. Thus, any function on instance graphs that does not decrease under these operations is admissible, since its value on I_p is greater than or equal to its value on $I_\mathbf{P}$. The proof is constructive: given any arbitrary instance graphs of a pattern and subpattern (as well as an arbitrary instance mapping function μ) we construct the sequence of operations leading from $I_\mathbf{P}$ to I_p, such that the mappingμ is not violated. □

Proof (details): Let $I_\mathbf{P} = (V_\mathbf{P}, E_\mathbf{P})$ and $I_p = (V_p, E_p)$, and let μ be the mapping from nodes of the instance graphs to the actual instances (subgraphs of the dataset D). We define the mapping $\text{SubInstances}: V_P \to 2^{V_p}$ as follows:

$$\text{SubInstances}(u) = \{v \in V_p \mid \mu(v) \subset \mu(u)\}$$

□

In other words, $\text{SubInstances}(u)$ is a set of all the subinstances of the instance u (actually, the above refers to the nodes in the respective instance graphs representing these instances). Similarly, let us define $\text{SuperInstance}: V_p \to 2^{V_\mathbf{P}}$ as follows:

$$\text{SuperInstances}(v) = \{u \in V_\mathbf{P} \mid \mu(v) \subset \mu(u)\}$$

Intuitively, $\text{SuperInstances}(v)$ is the set of all superinstances that contain a subinstance v.

In general, the sequence of operations will be to:

1. start from $I_\mathbf{P}$,
2. perform clique contractions and node additions in order to get the nodes V_p of graph I_p, and

3. perform edge deletions as necessary.

In the construction process, we will use the graph $I_{p\mathbf{P}} = (V_{p\mathbf{P}}, V_{p\mathbf{P}})$, initially equal to $I_{\mathbf{P}}$, and modify it, as well as construct a partial mapping λ from $I_{p\mathbf{P}}$ to I_p, such that at the end of the process λ will be an isomorphism. We also construct a *new instance mapping function* π and show that it is consistent with μ under the constructed isomorphism. (For conciseness, we do not index these mappings by the step number, because the mapping of a node never changes after it is first defined. For every instance graph node v we use$\pi(v) = \perp$, and $\lambda(v) = \perp$ to indicate that the respective mapping is currently undefined for v). Intuitively, π is a mapping from nodes of $I_{p\mathbf{P}}$ to the instances of the subpattern p in the database. Its "job" is to show that at the end of the process $I_{p\mathbf{P}}$ is indeed isomorphic to I_p.

As stated above, the sequence of operations begins with a sequence of clique contractions. During clique contraction steps we will also use a list of "marked" nodes (from $I_{\mathbf{P}}$) at each step, denoted **marked**, and a list of "covered" nodes from I_p, denoted **covered**. Construction proceeds as follows:

1. Let $I_{p\mathbf{P}} = I_{\mathbf{P}}$, **marked** $= \phi$ and **covered** $= \phi$, and for every node $v \in V_{\mathbf{P}}$ set $\pi(v) = \lambda(v) = \perp$.
2. Let u be a node in $I_p \backslash$**covered**, such that $V =$ SuperInstances$(u)\backslash$**marked** is non-empty. If there is no such node u, go to step 4.
3. (Note that the nodes V form a clique K in $I_{p\mathbf{P}}$, since each stands for a super-instance of the same u.)
   Let $k =$ Contract(K) in $I_{p\mathbf{P}}$ (changing $I_{p\mathbf{P}}$ as a side-effect), **marked** $=$ **marked** $\cup V$, **covered** $=$ **covered** $\cup \{u\}$, $\lambda(k) = u$, and $\pi(k) = \mu(u)$, and go to step 2.
4. For all u in $I_p \backslash$**covered**, do:

   (a). Add a unique new node v to $I_{p\mathbf{P}}$ using the operator $v = Add()$.
   (b). Let $\lambda(v) = u$, and $\pi(v) = \mu(u)$.

5. For every edge $e = \{v, v'\} \in I_{p\mathbf{P}}$ such that $\pi(v) \|_D \pi(v')$, do Remove(e) from $I_{p\mathbf{P}}$.

Note that the algorithm has considerable non-determinism in step 2, but any arbitrary selection of u at this step is sufficient. Clearly, $I_{p\mathbf{P}}$ is constructed from $I_{\mathbf{P}}$ using only the required operations.

Figure 3 illustrates the construction algorithm. In Fig. 3, (a) shows a (labeled, undirected graph) dataset, with a pattern **P** shown in (b) and a subpattern p shown in (c). The pattern **P** has 4 instances, three of which share at least one edge in common (hence the instance graph $I_{\mathbf{P}}$ shown in (d) contains a 3-clique), and one edge-disjoint instance. The subpattern p also has 4 instances, with an instance graph shown in (e). One instance mapping function consistent with the graphs is as follows (see 3(a) and (d)): $\mu(v_1)$ is the pattern including the node B at the bottom. $\mu(v_3)$ and $\mu(v_4)$ are the patterns consisting of the topmost triangle (T_4) and of the B nodes at the top, respectively. $\mu(v_2)$ is T_3, the second triangle from the top, plus the top-left B node. A consistent mapping for the subpattern would be (see 3(a) and (e)): $\mu(u_1)$ the bottom triangle (T_1), and $\mu(u_2)$ is T_2, the second triangle from bottom, $\mu(u_3)$ is T_3, the third triangle from the bottom, and $\mu(u_4)$ the top triangle, T_4. Note how each instance has an edge in common with other instances just as required by I_p.

Figure 3(f), (g) and (h) shows the instance graph transformations, as follows.

In step 2 of the algorithm, suppose that $u = u_4$ is chosen. Its superinstances are those represented by $V = \{v_3, v_4\}$. This clique is contracted in step 3, (with the node representing the clique being$k = v_3$) resulting in the graph shown in Fig. 3(f). At this step, u_4 is covered and $\{v_3, v_4\}$ are marked. At the next iteration, we might choose u_3 (which is then added

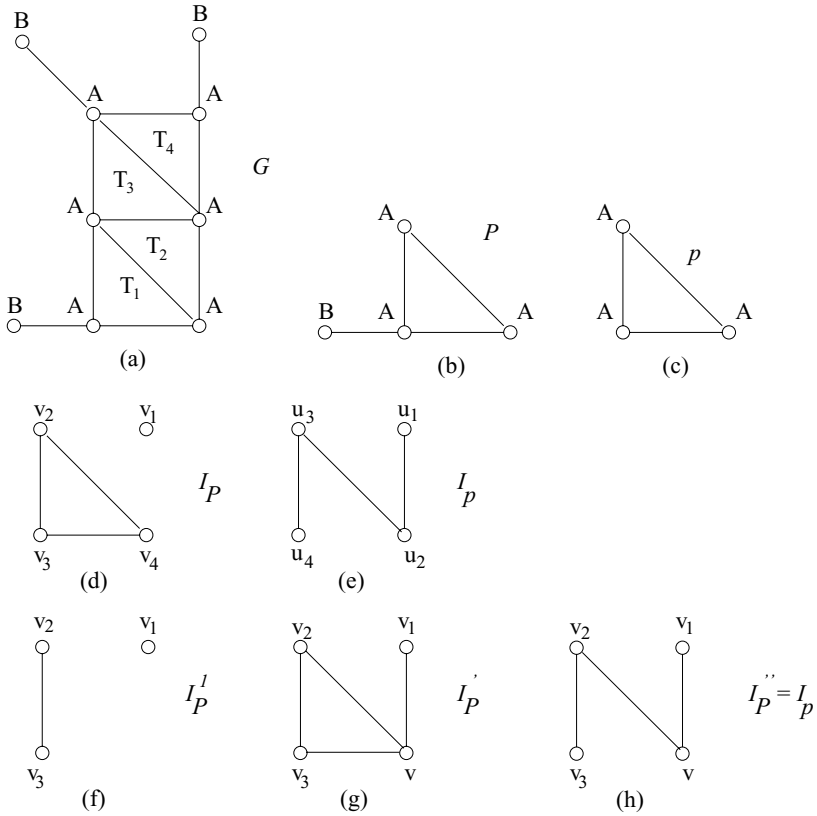


Fig. 3 Illustration of construction algorithm

to covered), with a superinstance v_2, which is marked. Contracting a singleton clique does not change the graph. At the final iteration, u_1 is selected (and added to covered), and its superinstance v_1 is marked and contracted (again, no change in the graph). There is no further iteration because u_2, the only uncovered node, has no superinstance, so we go to step 4.

In step 4, the only uncovered node is u_2, and the $v = \text{Add}()$ operation is done, resulting in the graph depicted in Fig. 3(g). Finally, in step 5, the edge at the bottom is removed, resulting in the instance graph of Fig. 3(h). Following the mappings generated in the construction will show consistency of the mappings, as well as the topology, to the instance graph I_p of Fig. 3(e).

We need to show that the construction algorithm is correct in the general case.

The following loop invariant can be shown (see Vanetik et al., 2005 for details):

Loop invariant (steps 2, 3): an edge $\{v, w\}$, consisting of nodes from $I_{\mathbf{P}}$, is in $I_{p\mathbf{P}}$ at the end of step 3 unless one of the following conditions occurs:

1. Either v or w or both are are not in $I_{p\mathbf{P}}$.
2. $\mu(w)$ and $\mu(v)$ have no edge intersections in D.
3. $\pi(v) \neq \perp$ or $\pi(w) \neq \perp$, but not both (w.l.o.g. let $\pi(w) \neq \perp$) and $\pi(w) \|_D \mu(v)$. (i.e. node v outside the clique was not mapped yet, and the superinstance represented by v does not intersect the subinstance represented by w.)

4. Both $\pi(v) \neq \perp$ and $\pi(w) \neq \perp$, and $\pi(w) \|_D \pi(v)$. (I.e. the two nodes were mapped to subinstances, but the subinstances do not intersect.)

The purpose of this invariant is basically to show that no required edges are lost during any of the iterations. There may be redundant edges, but these are immaterial, since edges can always be deleted later on by Remove() operations.

We now show that $I_{p\mathbf{P}}$ has the desired properties, i.e. that $I_{p\mathbf{P}}$ is an instance graph of pattern p in D, under the mapping π, and that it is isomorphic to I_p with the mapping λ being the required isomorphism. Let us first consider the nodes in $v \in V_{p\mathbf{P}}$:

Claim 1. For every instance g of p in D, there is a unique node v in $I_{p\mathbf{P}}$ such that $\pi(v) = g$, and a unique node u in I_p such that $\mu(u) = g$ and $\lambda(v) = u$. Additionally, $I_{p\mathbf{P}}$ contains no redundant nodes, i.e. for every $v \in V_{p\mathbf{P}}$ there exists an instance g of p in D suchthat $\pi(v) = g$.

Proof: By construction of $I_{p\mathbf{P}}$ (see Vanetik et al., 2005 for details). The following property of edges of $I_{p\mathbf{P}}$ follows immediately due to step 5: □

Claim 2. An edge $\{v, v'\}$ is in $I_{p\mathbf{P}}$ only if the instances $\pi(v), \pi(v')$ intersect.

Finally, we prove (see Vanetik et al., 2005) the following property of edge mapping:

Claim 3. For every distinct pair of instances g, g' of p, if $g \wedge g'$ then $\{\pi^{-1}(g), \pi^{-1}(g')\}$ is an edge in $I_{p\mathbf{P}}$.

Claims 1, 2, 3 show that $I_{p\mathbf{P}}$ is isomorphic to I_p, as required. Let f be function that satisfies the condition of the theorem. Since for every $p \subset \mathbf{P}$ the instance graph I_p can be obtained from $I_{\mathbf{P}}$ by using the three operations $A1, A2, A3$, and thus $f(I_p) \geq f(I_{\mathbf{P}})$, f is an admissible support measure. □

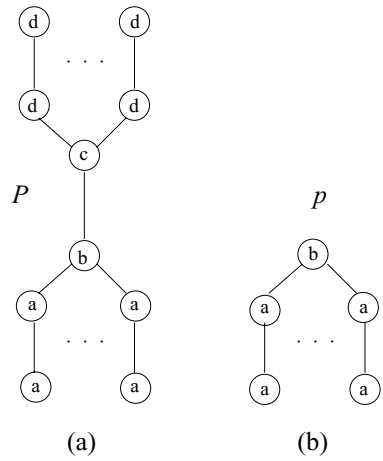
3.3. Necessary conditions for admissibility

In this section, we show that the above sufficient conditions for admissibility are also necessary, in the following sense.

Theorem 2. *Let f be positive real-valued function defined on pattern instance graphs. Then $f(I_{\mathbf{P}})$ is an admissible support measure* ***only if*** *it is non-decreasing under all of the operations: clique contraction, node addition, and edge removal on $I_{\mathbf{P}}$.*

Proof (outline): It is sufficient to show that there exists an instance graph, a required operation, a pattern and subpattern, and a dataset conforming to the instance graph and operation, to show that a support measure violating the conditions is inadmissible. However, we prove the theorem in a stronger sense. Given an arbitrary instance graph $I_{\mathbf{P}} = (V_{\mathbf{P}}, E_{\mathbf{P}})$, and an arbitrary required operation (clique contraction, node addition, or edge removal) on $I_{\mathbf{P}}$ (which results in a valid "sub-instance" graph I_p), we construct a pattern $\mathbf{P}$, a subpattern p, and a dataset D, such that $I_{\mathbf{P}}$ is the instance graph for $\mathbf{P}$ in D, and I_p is the instance graph for the subpattern p in D. Thus, if a support measure f ever decreases over any such operation (and for any instance graph), there will always be a dataset, pattern, and subpattern, for which the support of the subpattern is smaller than that of the superpatten, thus violating the admissibility of f. □

Fig. 4 Pattern and subpattern



Proof (details): In our construction, the form of the pattern and subpattern will actually be almost fixed, and parametrized only based on the size and degree of the instance graph. We begin by describing the patterns used in the construction. In order to simplify the proof, we will used labeled undirected graphs. □

3.3.1. Auxiliary graphs and their intersections

We construct the required dataset from labeled graph patterns that have the following structure.

Let $\delta = \max\{d(v) \mid v \in V_{\mathbf{P}}\}$. We define **P** to be the labeled graph pattern as in Fig. 4(a) with $|V_{\mathbf{P}}|$ edges labeled $\{a, a\}$ (the labels are actually on the vertices, but we refer to edges labeled by pairs of labels as a shorthand for referring to edges with incident nodes labeled by the indicated pair of labels) and δ edges labeled $\{d, d\}$. Let p be a subpattern of **P**, the subgraph induced by its $\{a, a\}$-edges and $\{a, b\}$-edges (see Fig. 4(b)). For convenience, we call each $\{a, a\}$ edge a **leg**, and each $\{d, d\}$ edge an **arm**. The subgraph consisting of all the edges $\{a, a\}$ and edges $\{a, b\}$ of a pattern is called a **bottom** (part of the pattern). We use the notation $Bottom(\mathbf{P})$ to denote the **bottom** subgraph of **P**. Likewise, the subgraph consisting of all the edges $\{d, d\}$ and edges $\{c, d\}$ of a pattern is called a **top** (part of the pattern). The corresponding function $Top(\mathbf{P})$ is used to denote the top subgraph of **P**. The edge $\{b, c\}$ in **P** is denoted by $Torso(\mathbf{P})$. Thus, **P** has a top (which in turn has δ arms), a torso, and a bottom (which in turn has $|V(I_{\mathbf{P}})|$ legs). The subpattern p has just the bottom part. The same terms will also be used to refer to such subgraphs of instances in the dataset D.

We define several ways in which instances will be allowed to overlap in the dataset D (these modes will also be called *intersection modes*). It is sufficient to explain these overlap types for two instances of **P** and/or p, denoted by G_1, G_2, and g_1 (see Fig. 5).

(I1) (Full) Bottom overlap: $Bottom(G_1) = Bottom\ (G_2)$. This type of intersection is used to indicate that G_1 and G_2 are adjacent in $I_{\mathbf{P}}$, but (since p is all bottom) the respective instances of p are all the same instance, thus correspond to a single node in I_p.

(I2) (Partial) Leg overlap: $Bottom(G_1)$ and $Bottom(G_2)$ overlap at exactly one leg (i.e. have a single common edge). This type of intersection indicates that G_1 and G_2 are adjacent in $I_{\mathbf{P}}$, and that the respective subpattern instances are distinct and adjacent in I_p.

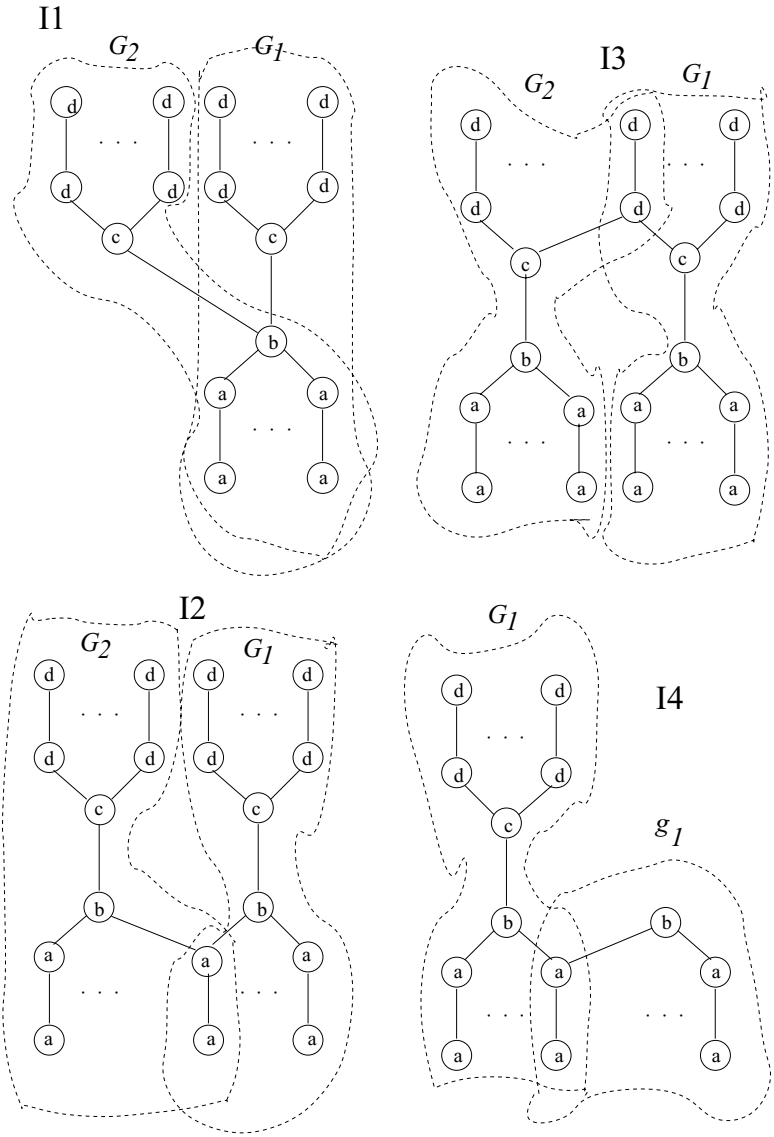


Fig. 5 Intersections

(I3) (Partial) Arm Overlap: $Top(G_1)$ and $Top(G_2)$ overlap at exactly one arm (i.e. have a single common edge). This type of intersection indicates that G_1 and G_2 are adjacent in $I_{\mathbf{P}}$, and that the respective subpattern instances are distinct and **not** adjacent in I_p. Such overlaps will be used to represent instance graph edges deleted by an operation.

(I1) Truncated (Partial) Leg Overlap: g_1 and $Bottom(G_1)$ overlap at exactly one leg (i.e. have a single common edge). This type of intersection indicates that g_1 is not a part of any superinstance, but its representation in I_p is adjacent to a node in I_p representing a subinstance that is part of some superinstance G_1.

During the construction of the dataset, all intersections of types I2, I3, I4 are subject to the **intersection condition:** for instances G_1, G_2, G_3 of $\mathbf{P}$ or p, if G_1 intersects G_2 at edge e, and G_2 intersects G_3 at edge f, then $e \neq f$. Note that the number of arms and legs of the patterns is specified as sufficiently large such that this condition can always be met in the construction. Additionally, the patterns and dataset are constructed such that no "unintended" instances appear. For a formal proof of these properties, see (Vanetik et al., 2005).

3.3.2. Construction of the dataset

In this section, we construct the dataset for any given instance graph $I_{\mathbf{P}}$ and any given required operation of type A1, A2, or A3. In all cases, the dataset D consists of $|V_{\mathbf{P}}|$ instances of $\mathbf{P}$, and the requisite number of instances of p. The construction differs mostly in the way the pattern instances are made to intersect. When constructing overlaps, the actual construction is arbitrary as long as the intersection condition is obeyed. For each of the three operation types, we construct the dataset D such that $I_{\mathbf{P}}$ is the instance graph of $\mathbf{P}$ in D, and that the I_p obtained by the operation is the instance graph of the sub-pattern p in D. Intuitively it is clear why the construction below should work. For a formal proof see (Vanetik et al., 2005).

Construction for A1—clique contraction. Let $I_{\mathbf{P}}$ be an arbitrary undirected graph. Let K be the contracted clique, V be the set of the nodes in contracted clique, and k be the node representing the clique (and remaining after contraction).

Let I_p be the undirected graph resulting from the operation $k = \text{contract}(K)$ in $I_{\mathbf{P}}$. The dataset D_{A1} consists of $|V_{\mathbf{P}}|$ instances of $\mathbf{P}$, such that $|V|$ of them (the ones corresponding to the contracted clique) intersect by (full) bottom overlap (this creates kind of a "star" fig.) The intersections among the other instances are determined by $I_{\mathbf{P}}$ as follows. Let $v, w \in V_{\mathbf{P}} \setminus V$. The instances represented by v and w have a leg overlap just when $\{v, w\} \in E_{\mathbf{P}}$. Now, partition the set of nodes $V_{\mathbf{P}} \setminus V$ into V_A, the set of nodes adjacent to k after the contraction (in $I_{\mathbf{P}}$, these nodes are adjacent to all nodes in V), and V_O, the set of nodes not adjacent to k after the contraction. For every $v \in V_A$, let the instance represented by v have a leg overlap with the instance represented by k. For every $v \in V_O$ and $w \in V$ such that $\{v, w\} \in E_{\mathbf{P}}$, let the respective instances have an arm overlap. (Intuition: consider an edge $\{v, w\} \in E_{\mathbf{P}}$. Clearly, if $v \in V_O$, then $\{v, w\}$ does not appear in E_p since arm overlap does not affect p, while if $v \in V_A$ the edge remains in E_p because a leg overlap indicates intersection in I_p as well.)

Construction for A2—node addition. Let $I_{\mathbf{P}}$ be an arbitrary undirected graph, $u = Add()$, and I_p be the graph resulting from this node-addition operation. The dataset D_{A2} consists of $|V_{\mathbf{P}}|$ instances of $\mathbf{P}$ (each also containing one instance of p), and one additional instance p that corresponds to u, with the intersections among instances determined by $I_{\mathbf{P}}$ as follows. For all $v, w \in V_{\mathbf{P}}$, the respective instances have a leg overlap just when $\{v, w\} \in E_{\mathbf{P}}$. The instance of p corresponding to u has a *truncated* (partial) leg overlap with all other instances. (Clearly the leg overlaps will result in the required edges in E_p, but the edges do not appear in $E_{\mathbf{P}}$ because there is no instance of $\mathbf{P}$ containing this instance of p.)

Construction for A3—edge removal. Let $I_{\mathbf{P}}$ be an arbitrary undirected graph, and I_p the graph resulting from Remove (e), with $e = \{v, v'\}$ the removed edge. The dataset D_{A3} consists just of $|V_{\mathbf{P}}|$ instances of $\mathbf{P}$ (each also containing one instance of p), with the intersections among instances determined by $I_{\mathbf{P}}$ as follows. For all $u, w \in V_{\mathbf{P}}$, the respective instances $\mu(u)$ have a leg overlap just when $\{u, w\} \in E_{\mathbf{P}} - \{e\}$, and an arm overlap just for the instances represented by v and v'. (Clearly an edge resulting from an arm overlap does not exist in E_p.)

The construction for the three cases completes the proof of Theorem 2, which also immediately implies the following potentially useful corollary:

Corollary 1. *Every graph is an instance graph, i.e for every graph G one can construct a dataset graph D and a pattern* **P** *such that G is the instance graph of* **P** *in D.*

4. Discussion

In this section, we examine non-trivial admissible support measures, compare related work, and discuss additional types of datasets (other than graphs) where our results apply. First, consider the support measure. As shown in Section 2.2, simply counting instances is not admissible. This is due to the fact that instances can share edges: since superpatterns have more edges than the subpattern, this potentially creates additional partially overlapped instances for the superpattern.

4.1. Independent set

The above observation on edge sharing between instances leads to the following intuition: count instances, but do not allow instances which overlap into the count. There are numerous ways to do this, but one obvious method is to find the a set of non-overlapping instances, and count its size. Since the instance graph contains all information about the instances and their overlaps, it is sufficient to define this type of support function over the instance graph. A set of non-overlapping instances maps uniquely to an independent set in the instance graph—i.e. a set of vertices in the instance graph, none of which are connected by an edge in the instance graph.

One particular such measure is the size of the largest independent set in the instance graph, i.e. the MIS measure. Our results justify using the maximum independent set size as a support measure.

Corollary 2. *Maximum independent set size of* $I_{\mathbf{P}}$ *is an admissible support measure.*

Proof: Clearly, edge removal does not decrease the size of the maximum independent set of a graph, and neither does node addition, since no edges between existing nodes in the graph are added. Thus, it suffices to show that clique contraction cannot decrease this size. Let s be an independent set of maximum size in $I_{\mathbf{P}}$, and let K be a clique in $I_{\mathbf{P}}$. Observe that clique contraction adds no edges to nodes outside of K (other than to k, the "contracted clique" new node). Thus, if K contains no nodes from s, then s is still an independent set after the operation. Alternately, K contains exactly one node v from s (there cannot be more than one node of K from s). None of the neighbors of v are in s, and thus, by construction of the clique contraction operator, none of the neighbors of k after the operation are in s. Since no edges except some incident on k are added by the operator, then $(s \setminus \{v\}) \cup \{k\}$ is an independent set after the operation, and its is size is not less than $|s|$, as required. □

It is unfortunate that the problem of determining the maximum independent set size in a graph is NP-hard, and hence independent set size is not very efficient as a support measure in the worst case. However, in practice, that is not likely to be a problem, for several reasons:

1. The computational cost of finding the instances in a large dataset may be greater than that of finding the independent set, and therefore other tasks that are presumed hard in the worst case (in particular, finding an a subgraph isomorphism) may overshadow the support computation.
2. The topology of the instance graph for realistic datasets and candidate patterns is unlikely to result in a hard problem instance (see below). This was also shown in the experimental evaluation on both synthetic and real-life graphs as reported in Vanetik et al. (2004) and Vanetik and Gudes (2004).
3. In many cases, support is only computed in a graph mining algorithm in order to decide whether or not the candidate pattern is frequent (and if not, the pattern is discarded anyway). An approximation or a bound of the independent set size may well be sufficient to make a decision (for example, finding one independent set of size higher than the threshold means that the pattern is frequent, regardless of the size of the maximum independent set). Such an approximation is also used by Kuramochi and Karypis (2004).

Observe that this support measure clearly reduces to the standard support measure of transaction databases, since transactions are considered independent from each other, and the maximum independent set size simply equals the number of transactions containing the pattern.

4.2. Other support measures

Another intuitive support measure is *weighted instance count*. The idea is to count each instance only partially, depending on how many overlaps it has with other instances. Formally, in terms of the instance graph, this measure is: weighted sum over the nodes in the instance graph, where the weight of each node is a reciprocal of its degree $d(v)$, i.e.:

$$S_1(I) = \sum_{v \in I_{\mathbf{P}}} \frac{1}{d(v)+1}$$

In our early graph-mining research we initially attempted to use the measure S_1, but later on suspected that it was not admissible. Nevertheless, we had to work rather hard to come up with a counter-example in the form of a pattern, a sub-pattern, and a dataset. Using our main result, it is sufficient to show a case where S_1 decreases under node addition. This is easily done—the simplest such case is where $I_{\mathbf{P}}$ consists of two nodes connected by an edge, thus $S_1(I_{\mathbf{P}}) = 2$, but after adding a new node and its edges we have $S_1 = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} < 2$ (see Fig. 6). Hence, S_1 decreases under node addition and is thus not admissible. Obviously, our results also imply that no monotonic function of S_1 can be an admissible support easure. Note that the MIS measure is 1 in both cases, thereby maintaining admissibility.

Fig. 6 Weighted count measure

4.3. Other applications of our results

Several graph-mining problems restrict the topology of either patterns, or datasets, or both. Observe that all of our results hold for the case where the patterns are restricted to be tree-shaped, as long as the datasets can be general graphs, because our proofs only used tree-shaped patterns. If we restrict datasets to be tree-shaped as well, our sufficient conditions would still hold, but the necessary conditions may no longer apply.

Other possible graphs of interest are unlabeled graphs, directed graphs, and alternate definitions of the instance graph. First, consider the sufficient conditions for admissibility. Here, the only property of the pattern and dataset graphs that we used was: if instances of subpattern intersect, then instances of the superpattern also intersect. Clearly this property holds w.r.t. edge intersection for both directed and unlabeled graphs (and also for directed unlabeled graphs). Thus, our sufficient conditions hold for these types of graphs as well.

Allowing a different definition of instance graph makes the issue a bit trickier. For example, redefine intersection between instances to be: instances intersect if they have at least one **node** in common. Note that the resulting instance graph will then be denser than an instance graph defined by edge intersection, because edge intersection implies node intersection, but not vice-versa.

Open question: Do the *sufficient* conditions for admissibility hold for instance graphs based on node intersection?

As to the necessary conditions for admissibility, again our result can be extended. For directed labeled graphs, simply direct the edges in the pattern and dataset in any arbitrary but consistent manner, for example from top to bottom. The construction and proof are unchanged. For unlabeled graphs the situation is trickier, since the pattern must be considerably enlarged (e.g. by encoding labels as part of the graph topology) to ensure that no spurious instances appear, but it should be possible. The directed unlabeled case is somewhat easier, because edge directions allow more structure that prevents spurious instances in the construction. For instance graphs based on node intersection, we believe that the necessary conditions hold, but have no proof:

Conjecture: The *necessary* conditions for admissibility hold for instance graphs based on node intersection.

Another possible application for our results is for data mining tasks other than graph mining, but such that instance relationships and support can be stated in terms of instance graphs. Observe that this is trivially the case for itemsets in transaction databases, where the instance graph has no edges. However, a non-trivial mapping would occur in data mining in the face of abstractions or generalizations (Srikant and Agrawal, 1995), when considering support measures that may count more than one occurrence within a single transaction.

4.4. Related work

Although the support measure is a central issue in data mining, there are few attempts to define properties of support generally as is done in this paper. As mentioned before, the common support measure is defined for transaction databases as the ratio of the number of transactions containing an itemset to the total number of transactions in the database Agrawal and Srikant (1994). Obviously, this measure is admissible.

In Chen et al., (1998) define a support measure for mining web traversals in a form of trees. Each transaction is a tree, and the support of a subtree is defined as the number of transactions containing the subtree, but with the restriction that only transactions with trees which are not included in other transactions are counted (this is a very special restriction

which is quite difficult to generalize). It is easy to show that because of the special structure of trees, this measure is admissible.

Wang and Liu (1998) discuss the mining of frequent structures of documents in another form of trees. They are interested only in rooted tree patterns and define support as the number of occurrences of such (maximal) trees in a set of documents. Again, this measure is obviously admissible.

The most related papers on graph mining are Kuramochi and Karypis (2001) and Yan and Han (2002). In both papers, the support is defined as in transaction databases. Each transaction is a graph, and the support is defined as the number of transactions containing the pattern graph (no matter how many times). Again, it is obvious that this support measure is admissible. It is also obvious that this measure is not appropriate for a dataset defined as a single graph, because such a support measure can only result in values of either 0 or 1 in this case, no matter how many times the simple pattern occurs in the large graph. Earlier papers on graph mining and their applications are (Chen et al. 1998; Lin et al., 1998; Wang and Liu, 1998; Kuramochi and Karypis, 2001; Yan and Han, 2002). Paper (Wang et al., 2002) uses a hashing approach, first introduced in Pennec and Ayache (1998), to obtain an approximate number of a subgraphs (unique up to several graph-modifying operations) matching the given pattern.

The only other work which uses other notions of support that we know of in the context of graph mining, besides our own (Vanetik and Gudes, 2004; Vanetik et al., 2002; Vanetik et al., 2004; Vanetik, 2002), is that of Kuramochi and Karypis (2004) which was published later. There, the concept of single graph setting is defined, and the paper proposes to use the measure which we presented in Vanetik et al. (2002), namely the maximum independent set measure. An interesting contribution of Kuramochi and Karypis (2004) is that the value of the support measure may be dynamic or adaptive, but the definition itself does not change. As far as we know, the current paper is the first attempt to define a general notion of support, especially for graph mining, and prove its admissibility and related properties.

5. Conclusions

In this paper we discussed a general notion of support especially useful for mining of graph datasets and databases. We defined the concept of admissible support measures and proved sufficient and necessary conditions for admissibility. An intuitive measure (size of maximum independent set) was presented and shown to be admissible. This latter measure was already used in two other papers on graph mining. Future work includes finding other instances of admissible support measure which are of interest for different classes of graphs.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. Proc. of the 20th Int'l Conf. on VLDB, Santiago, Chile

Bray T, Paoli J, Sperberg-McQueen C, (Eds.) (1998) Extensible Markup Language (XML) 1.0, February, http://www.w3.org/XML/#9802xml10

Chamberlin D (2003) XQuery: A query language for XML, Proceedings of SIGMOD Conference

Chen MS, Park JS, Yu PS (1998) Efficient data mining for path traversal patterns. IEEE Transactions on Knowledge and Data Engineering 10(2):209–221

Dehaspe L, Toivonen H, King RD (1998) Finding frequent substructures in chemical compounds. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98) New York, New York, pp. 30-36

Deutsch A, Fernandez M, Florescu D, Levy A, Maier D, Suciu D (1999) Querying XML data. IEEE Data Engineering Bulletin 22(3):27–34

Deutsch A, Fernandez MF, Suciu D (1999) Storing semistructured data with STORED. Proceedings of SIGMOD Conference, pp. 431–442
Domshlak C, Brafman R, Shimony SE (2001) Preference-based configuration of web page content. Proceedings of IJCAI
Goldman R, Widom J (1997) DataGuides: Enabling query formulation and optimization in semistructured databases. Proc. of 23rd VLDB Conf., Athens, Greece
Graph Matching Library, http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html
Yan X, Han J (2002) gSpan: Graph-based substructure pattern mining. Proceedings of ICDM, pp. 721–724
Huffman SB, Baudin C, Toward structured retrieval in semi-structured information spaces, Proceedings of IJCAI-97, Nagaya, Japan, pp. 751–756
Inokuchi A, Washio T, Motoda H (2000) An apriori based algorithm for mining frequent substructures from graph data. Proceedings of PKDD00
Kuramochi M, Karypis G (2004) Finding Frequent Patterns in a Large Sparse Graph Proceedings 2004 SIAM Data Mining Conference, Orlando, Florida
Kuramochi M, Karypis G (2001) Frequent subgraph discovery. Proceedings of IEEE ICDM
Lin X, Liu Ch, Zhang Y, Zhou X (1998) Efficiently computing frequent tree-like topology patterns in a web environment. Proceedings of 31st Int. Conf. on Tech. of Object-Oriented Language and Systems
Maximum weight clique program, http://www.tcs.hut.fi/ pat/wclique.html
McKay BD (1998) Isomorph-free exhaustive generation. Journal of Algorithms 26:306–324
Meisels A, Orlov M, Maor T (2001) Discovering associations in XML data. BGU Technical report
Milner R (1983) Calculi for synchrony and asynchrony. Proceedings of TCS 25:267–310
Ng RT, Lakshmanan LVS, Han J, A. Pang (1998) Exploratory mining and pruning optimizations of constrained association rules. Proceedings of SIGMOD Conference, pp. 13–24
Movie database, http://us.imdb.com
Ostergard PRJ (2001) A new algorithm for the maximum-weight clique problem, Helsinki University of Technology, internal report
Pennec X, Ayache N (1998) A geometric algorithm to find small but highly similar 3D substructures in proteins. Bioinformatics 14(6):516–522
Srikant R, Agrawal R (1995) Mining generalized association rules. Proceedings of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland
Vanetik N (2002) Discovery of frequent patterns in semi-structured data. M.Sc. thesis. Dept. of Computer Science, Ben Gurion University
Vanetik N, Gudes E (2004) Mining frequent labeled and partially labeled graph patterns. Proceedings of ICDE, Boston, pp. 91–102
Vanetik N, Gudes E, Shimony SE (2002) Computing frequent graph patterns from semistructured data. Proceedings ICDM, pp. 458–465
Vanetik N, Shimony ES, Gudes E (2004) Computing frequent graph patterns using disjoint paths. submitted for a journal publication
Vanetik N, Gudes E, Shimony SE (2005) Support measures for graph data. Technical Report FC-06-02, Computer Science Dept., Ben Gurion University
Wang K, Liu H (1998) Discovering Typical Structures of Documents: A Road Map Approach. Proceedings of SIGIR, pp. 146–154
Wang X, Wang JTLi, Shasha D, Shapiro B, Rigoutsos I, Zhang K (2002) Finding patterns in three-dimensional graphs: Algorithms and applications to scientific data mining. IEEE Trans on Knowledge and Data Eng 14(4):731–749
Washio T, Motoda H (2003) State of the art of graph-based data mining. SIGKDD explorations