Journal of Computer Science 3 (9): 708-716, 2007
ISSN 1549-3636
© 2007 Science Publications

Support Measure Admissibility Based on Operations on Instance Graphs

Ford Lumban Gaol
Faculty of Computer Science, University of Indonesia, Indonesia

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 grater than that of its subpatterns, a property called anti-monotonicity or admissibility. This study examines the requirements for admissibility of a support measure. Support measure 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 ad 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.

Key words: Data mining, frequent patterns, graphs, support measure

INTRODUCTION

The primary goal of data mining is to discover interesting patterns in data. Since patterns that appear frequently may 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? (we use the term dataset through the paper, to cover all sorts of data, including data in relational database, semistructured databases, web data, etc.). The answer of this question is then used to decide whether a pattern is interesting, either individually or in relation to other patterns. Usually the frequency of a patterns P is called support of P (in D). An early appearance of this was in the classical paper by Agrawal et. all on mining association rules[1] .

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 transaction. The standard measure support for itemsets in the literature is as follows. Let D be a set of transactions, and
I = $<i_1,.., i_k>$ be an item set. The support S of the itemset in the dataset is defined as

$$S(I) = \frac{|\, t \mid t \in D, < i_1,\ldots,i_k > \in t \,|}{|D|}.$$

Usually in data mining tasks, a number $0 \le \sigma \le 1$, called the minimum support threshold is provided to the system. An itemset I that has $S(I) \ge \sigma$ is called frequent. Defined in this manner, the support of an itemset 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.*,[21]) has been of major importance in numerous data mining algorithms , used to prune candidate patterns and greatly improve performance. This property is the de-facto standard assumption for algorithms, which relay heavily on anti-monotonicity. These algorithms range from the apriori algorithm for structured data[1] to algorithms to mining paths[4], trees[27] and graphs[15].

Since data sets in many applications are not limited to transaction database, and patterns are mot 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 suppor*t* 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 $S: D \times P \rightarrow \Re_+ \cup \{0\}$ 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(P) is greater than or equal to a support threshold TS. 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 measure (i.e. anti-monoticity or downward closure) for itemsets indicates its general applicability, leading to the following definitions.

Definition 1 (Admissible support measure): A support measure S is admissible if for every dataset D and pattern P we have $S(P) \geq 0$, and f or every pattern P' ∍ $P' \subseteq P$ (meaning that P' is a subpattern of P) we have $S(P') \geq S(P)$.

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[26] such as web data[12, 27, 3], chemical compounds data[28] or biological data[24]. Although for itemset 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 and in object databases, topology is meaningful, a data mining task that has been 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 show 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 and sufficient 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 naïve 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. The use of MIS as support measure was first suggested in[28] and was show to be useful as a major component of an apriori-based algorithm for graph mining. The major contribution of the present paper is the formal definition and 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 graph, examines the intuitive definitions of support for graphs, and defines the useful admissible MIS support measure. Section 3 defines operations on instance graphs, and use them to show our main result, the necessary and sufficient conditions for admissibility. Section 4, shows that the MIS measure is admissible, examines other support schemes and discuss generalizations of our results. Finally, the related work is examined and compared to ours.

INSTANCE GRAPHS AND BASIC SUPPORT MEASURES

We begin with assumption 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) unlabelled graphs, as these are equivalent to a special case where $|\Sigma| = 1$.

Instances and instance graphs: Let D be a (not necessarily connected) labeled dataset graph, and P be the pattern for which we are searching – a connected labeled graph.

Definition 2:. Subgraph G of D is an instance of P in D just when there exists a label-preserving isomorphism between P and G.

Note that according to definition 2, instances of 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 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 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 P its instance graph in D, denoted $I_P(D)$ is an undirected graph (V_P, E_P), with exactly one node standing for each

Notation used in this paper

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

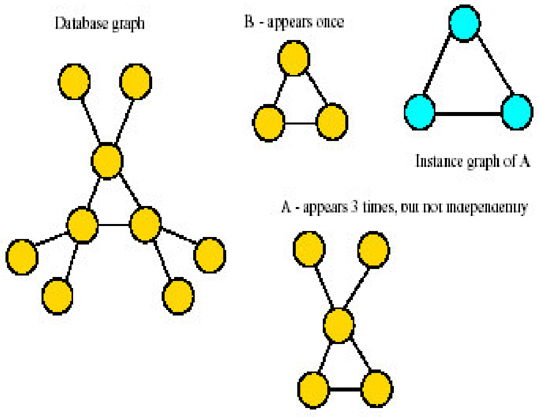

Fig. 1: Graph pattern support

instance of P in D. denoted by μ the mapping from nodes in V_P to instances (subgraphs of D). The edges in the instance graph are $E_P = \{\{u, v\}| \mu (u) \wedge \mu (v)\}$. 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_P to denote $I_P(D)$.

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

In the rest of the paper, P and p denote respectively a pattern and its subpattern in D. By definition, there exits a mapping μ are implicitly understood, we denote the subgraph relation $\mu(v) \subset \mu$ (u) between instances in D, by $v \subset u$.

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 pattern P is never strictly greater than the support of any of its subgraphs.
The most intuitive support measure is the number of instances of the pattern in a dataset. This measure, however, is not admissible.

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 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 fulfils 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 admisitability.

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

Definition 4: (Clique contraction) Clique contraction operates on a subgrpah of G forming a clique = (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 nodes $V' = V \backslash V_K \cup \{k\}$, and the edges: $E' = (E \cap \{\{u, v\}|u, v \in V'\}) \cup \{\{v, k\}|v \in V' - \{k\} \wedge \forall u \in V_K \{v, u\} \in E\}$.
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 *K* in *G*. This definition applies to any clique, not necessarily a maximum clique. For the sake of uniformity, a subgraph

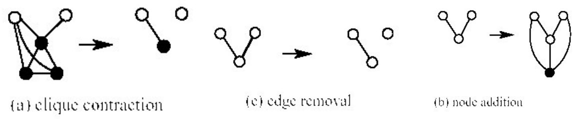

Fig. 2: Examples of Operations on Instances Graphs

consisting of a single node is considered to be a clique, and in this case clique contraction does not change the graph.

Definition 5: (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 $E' = E \cup \{\{u, v\} | v \in V\}$.

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

Definition 6: (Edge removal) Edge removal removes an edge e from G. The resulting graph is $G' = (V, E \backslash \{e\})$.

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 treat two node differently based on any node labels or identity, such as the identity of the instances the node represent in the dataset).

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

Theorem 1: A positive valued function $f(I_P)$ is an admissible support measure over graph patterns IF it is non-decreasing under all of the following operations on I_P:

(A1) Clique contraction: k = Contract(K), where K is a clique , and k is the node representing the clique after the contraction operator.

(A2) Node addition: v = Add(), where v is the new node added by the operator.

(A3) Edge removal: Removal(e), where e is an edge to be removed.

Proof: We prove that I_p can be obtained from I_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_P. The proof is constructive: given any arbitrary instance graphs of pattern and subpattern (as well as an arbitrary instance mapping function μ) we construct the sequence of operation leading from I_P to I_p such that the mapping μ is not violated.

Let $I_P = (V_P, E_P)$ and $I_p = (V_p, E_p)$ and let μ be the mapping from node of the instance graphs to the actual instances (subgraphs of the dataset D). We define the mapping SubInstances: $V_P \rightarrow 2^{V_p}$ as follows:
SubInstances(u) = $\{v \in V_p \mid \mu(v) \subset \mu(u)\}$
In other words, 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 SuperInstance: $V_P \rightarrow 2^{VP}$ as follows:
SuperInstances(v) = $\{u \in V_P \mid \mu(v) \subset \mu(u)\}$
Intuitively, SuperInstance(v) is the set of all Superinstances that contain a Subinstance v.
In general, the sequence of operations will be to:
start from I_P,
perform clique contractions and node additions in order to get the nodes V_p of graph I_p, and
perform edge deletions as necessary.

In the construction process, we will use the graph $I_{pP} = (V_{pP}, E_{pP})$, initially equal to I_P, and modify it, as well as construct a partial mapping 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 π (v) = ⊥, and λ(v) = ⊥ to indicate that the respective mapping is currently undefined for v). Intuitively, π is a mapping from nodes of I_{pP} to the instances of the subpattern p in the database. Its "job" is to show that at the edn of the process I_{pP} is indeed isomorphism 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_P) at each step, denoted marked, and a list of "covered" nodes from I_p denoted covered. Construction proceeds as follows:

- Let $I_{pP} = I_P$, marked = ϕ and covered = ϕ, and for every node $v \in V_P$ set $\pi(v) = \lambda(v) = \perp$.
- Let u be a node in I_p \ covered, such that V = SuperInstances(u) \ marked is non-empty. If there is no such node u, go to step 4.
- (Note that the nodes V from a clique K in I_{pP}, since each stands for a Superinstances of the same u).
  Let k = Contract(K) in I_{pP} (changing I_{pP} as a side-effect),
  marked = marked ∪ V, Covered = Covered ∪ {u}, λ(k) = u, π(k) = μ(u), and go to step 2.

- For all u in I_p\Covered, do:
  (a). Add a unique new node v to I_{pP} using the operator v = Add().
  (b). Let $\lambda(v) = u$, and $\pi(v) = \mu(u)$.
- For every edge e = {v, v'} ∈ I_{pP} such that $\pi(v) \|_D \pi(v')$, do Remove(e) from I_{pP}.

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

(Fig. 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 leas one edge in common (hence the instance graph I_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)): $\mu(u_1)$ is the pattern including the node B at the bottom. $\mu(u_3)$ and $\mu(u_4)$ are the patterns consisting of the topmost triangle (T_4) and of the B nodes at the top, respectively. $\mu(u_2)$ is T_3, the second triangle for 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 tringle (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.

(Fig. 3) (f), (g), and (h) shows the instance graph transformation, 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

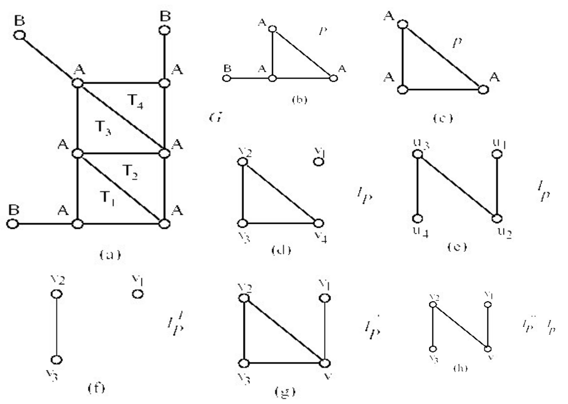

Fig. 3: Illustration of construction azgorithm

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 to covered), with a superintance 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 superintance 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 = 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 mapping 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).

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_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_P

Proof: (outline): It is sufficient to show that there exist an instance graph, a required operation, a pattern and subpattern, and a dataset confirming 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_P = (V_P, E_P), and an arbitrary required operation (clique operation, node addition, or edge removal) on I_P, (which results in a valid "sub-instance" graph I_p), we construct a pattern P, a subpattern p, and a dataset D, such that I_P is the instance graph for P in D, and Ip is the instance graph for the subpattern p in D. Thus, if a support measure f ever decrease 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 superpattern, thus violating the admissibility of f.

3.1. Auxiliary graphs an their intersections:

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

Let $\delta = \max \{d(v) | v \in V_P\}$. We define P to be the labeled graph pattern as in Fig. 4(a) with $|V_P|$ edges labeled {a, a} (the labels are actually on the vertices, but we refer to edges labeled by pairs pf labels as a shorthand for referring to edges with incident nodes labeled by their 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 edges {a, a}and edges {a, b} of pattern is called a bottom (part of the pattern). We use the notation Bottom(P) to denote the bottom subgraph of P. Likewise, the subgraph consisting of all the edge {d, d} and edges {c, d} of pattern is called a toP (part of the pattern). The corresponding function Top(P) is used to denote the top of subgraph P. The edge {d, c} in P is denoted by Torso(P). Thus, P has a top (which in turn has δ arms), a torso, and a bottom (which in turn has $|V(I_P)|$ legs). The subpattern p has just the bottom part. he same terms will also be sued to refer to such subgraphs of instances n the dataset D.

We define several ways which instances will be allowed to overlap in the dataset D (these nodes 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_P, but (since p is all bottom) the respective instances of p are all the same instances, thus correspond to a single node in I_p.

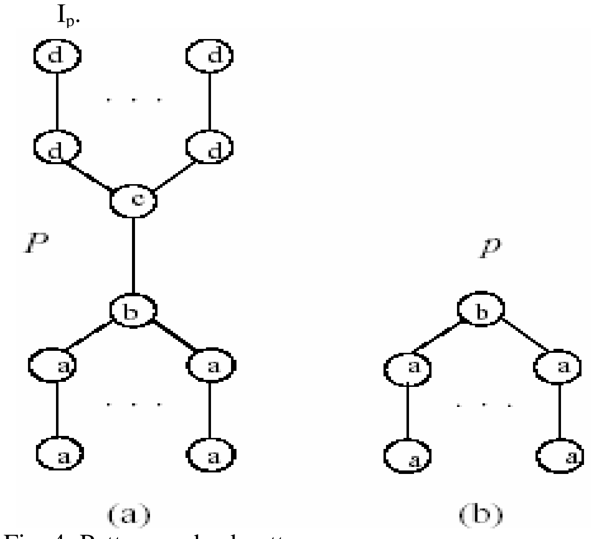

Fig. 4: Pattern and subpattern

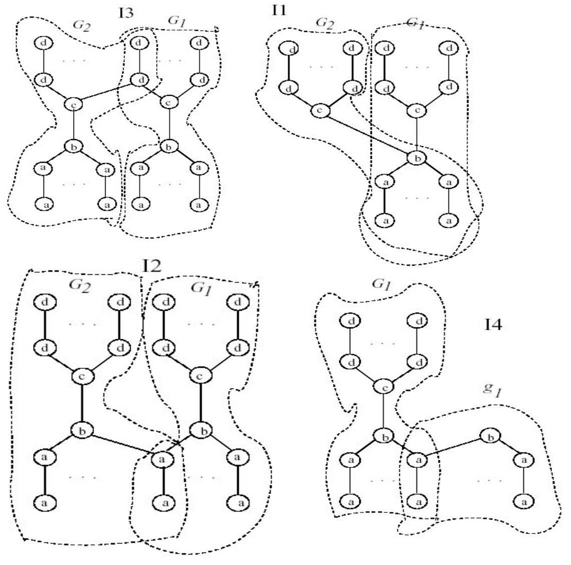

Fig. 5: Intersections

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_P, and the respective subpattern instances are distinct and adjacent in I_p.

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_P, and that respective subpattern instances are not adjacent in I_p. Such overlaps will be used to represent instance graph edges deleted by an operation.

I4: 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 superinstances, but its representation in I_p is adjacent to a node in I_p representing a subinstance that is part of some superintance G_1.

During the construction of the dataset, all intersections of types I2, I3, 4 are subject to the intersection condition: for instances G_1, G_2, G_3 of P or p if G_1 intersects G_2 at edge e, and G_2 intersects G_3 at edge f, the $e \neq f$. 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.

3.2. Construction of the dataset: In this section, we construct the dataset for any given instance graph I_P and any given required operation of type A1, A2, or A3. In all cases, the dataset D consist of $|V_P|$ instances of P, and the requisite number if instances of p. The constructive 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_P is the instances graph of P in D, and that the I_p is obtained by the operation is the instance graph of the subpattern p in D. Intuitively, it is clear why the construction below should work.

Construction for A1-clique contraction : Let I_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 = contract (K) in I_P. The dataset D_{A1} consists of $|V_P|$ instances of 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_P as follows. Let v, w $\in V_P\backslash V$. The instances represented by v and w have a leg overlap just when $\{v, w\} \in E_P$. Now, partition the set of nodes $V_P\backslash V$ into V_A, the set of nodes adjacent to k after the contraction (in I_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 $\{v, w\} \in E_P$, let the respective instances have an arm overlap. (Intuition: consider an edge $\{v, w\} \in E_P$. Clearly, if $v \in V_O$, the $\{v, w\}$ does not appear in E_p since arm overlap does not affect p, while if $v \in VA$ the edge remains in E_p because a leg overlap indicates intersection in I_p as well).

Construction for A2-node addition: Let I_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 $|V_P|$ instances of 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_P as follows. For all v, $w \in V_P$, the respective instances have a leg overlap just when $\{v, w\} \in E_P$. 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_P because there is no instance of P containing this instance of p).

Construction for A3-edge removal: Let I_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 $|V_P|$ instances of P (each also containing one instance of p), with the intersection among instances determined by I_P as follows. For all u, $w \in V_P$, the respective instances $\mu(u)$ have a leg overlap just when $\{u, w\} \in E_P - \{e\}$, 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 potential 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*.

DISCUSSION

In this section, we examine non-trivial admissible support measure, 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 superpattern have more edges than the subpattern, this potentially creates additional partially overlap instances for the superpattern.

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 set of non-overlapping instances, and count its size. Since the instances 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 large 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_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 can not decrease this size. Let s be an independent set of maximum size in I_P, and let K be a clique in I_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 can not be more than one node of K from s). None of the neighbors of v are in s, and thus, by contraction 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 $(s\backslash\{v\})\cup k\}$ 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 graph is NP-hard, and hence independent set size is not very efficient as a support measure.

Although the support measure is a central issue in data mining, there are few attempts to define properties of support generally. 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 transaction in the database[1]. Obviously, this measure is admissible. In[4] Chen et al. define a support measure for mining web transversal 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 transaction with trees which are not included in other transaction 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.

The mining of frequent.[27] structures of documents in another form of trees. They are interested only in rooted tree patterns and define suppot as the number if 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[15] and Han[11]. In both papers, the support is defined as in transaction databases. Each transaction 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 in graph mining and their applications are[4, 16, 27, 15, 11].

CONCLUSION

In this paper we discus a general notion of support especially useful for mining of graph dataset and database. We defined the concept of admissible support measure and proved sufficient and necessary conditions for admissibility. An intuitive measure (size of maximum independent set) was presented and show to be admissible. Future work includes finding other instances of admissible support measure which are of interest for different classes of graphs.

REFERENCES

1. Agrawal R. and R. Srikant. Fast Algorithms for Mining Association Rules. Proc of 20th Int'l Conf. on VLDB. Santiago, Chile September 1994.
2. Bay, T., J. Paoli, and C. Sperberg-McQuen. Extensible Markup Langauge (XML) 1.0. February 1998. http://www3.prg/XML .
3. Chamberlin, D., Xquery: A Query Language for XML. Proceedings of SIGMOD Conference 2003.
4. Chen, M.S., J.S. Park, P.S. Yu. Efficient Data Mining for Path Transversal Patterns. IEEE Transactions on Knowledge and Data Engineering. 10(2). 1998: 209-221.
5. Dehaspe, L., H. Toivonen and R.D. King. Finding Frequent Substructure in Chemical Compounds. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98) pages 30-36. New York.

6. Deutsch, A. et. al. Querying XML data., IEEE Data Engineering Bulletin 22(3). 1999.
7. Deutsch, A. et. al. Storing Semistrucred Data With STORED. Proceedings of SIGMOD Conference 1999: 431-442.
8. Domshalk, C. et al. Preference-based Configuration of web Page Content. Proceedings of IJCAI, August 2001.
9. Goldman, R. et al. DataGuides: enabling Query Formulation and Optimization in Semistrcured Databases. Proc. of 23rd VLDB Conf., Athens Greece 1987.
10. Graph Matching Library, http://amalfi.dis.unima.it/graph/db/vflib-2.0/doc/vflib.html
11. X.Yan and J.Han, gSpan: Graph Based substracture pattern mining. Proceedings of ICDM 2002, pp 721-724.
12. S.B. Huffman, C. Baudin, Toward Structure Retrieval in Semistructured Information Spaces, in Proceedings of IJCAI-97, Nagaya, Japan: 751 – 756
13. A. Inokuchi, T. Washio, H. Motoda. An Apriori based algorithm for mining frequent substructure from graph data. Proceedings OF PKDD00. 2000.
14. M. Kuramochi & G. Karypis. Finding Frequent Patterns in a Large Sparse Graph. Proceedings 2004 SIAM Data Mining Conference, Orlando, Florida, 2004.
15. M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. Proceedings of IEEE ICDM 2001.
16. X. Lin, Ch. Liu, Y.Zhang and X.Zhou. Efficiently Computing Frequent Tree-Like Topology Patterns in a Web Environment.. Proceedings Of 31st Int. Conf. on Tech. of Object Oriented Language and System, 1998.
17. Maximum Weight Clique Program http://www.tcs.hut.fi/~pat/wclique.html.
18. B.D. Mackay. Isomorph-free exhaustive generation. Journal of Algorithms, Vol 26, 1998:306-324.
19. A. Meisels, M.Orlov, T. Maor. Discovering Associations in XML data. BGU Technical Report.. 2001.
20. R. Milner. Calculi for synchrony and asynchrony. Proceedings of TCS 25, 1983:2677-310.
21. R.T. Ng. L.V.S. Laksmanan. J. Han, A. Pang: Exploratory Mining Pruning Optimization of Constrained Associations Rules. Proceedings of SIGMIOD Conference 1998: 13-24.
22. Movie databse. http://us.imdb.com.
23. P.R.J. Ostergard. A new algorithm for the maximum-weight clique problem. Helsinki University of Technology. Internal report 2001.
24. X.Pennec, N. Ayache. A geometric algorithms to find small but highly similar 3D substructures in protein. Bioinformatics 14:(6): 516-522. 1998.
25. R. Srikant, R. Agrawal. Mining Generalized Association Rules. Proceedings of the 21st Int'l Conference on Very Large Database. Zurich Switzerland. Sept 1995.
26. Ford Lumban Gaol, Belawati Widjaja. Mining Frequent Semistructure Pattern Using Path Covers. The 2nd Indonesia Japan Joint Scientific Symposium, 2006. Accepted .
27. K. Wang , H. Liu. Discovering Typical Structures of Documents: A Road map approach. Proceedings of SIGIR 1998: 146 – 154.
28. X. Wang, J.T.Li et al. Finding patterns in Three-dimensional Graphs Algorithms and applications to scientific Data Mining. IEEE Trans on Knowledge and Data Eng 14(4): 731-749. 2002.
29. T. Washio, H. Motoda. State of the art of graph-based data mining. SIGKDD explorations. July 2003.