An Efficiently Computable Support Measure for Frequent Subgraph Pattern Mining

Yuyi Wang and Jan Ramon

Department of Computer Science
Katholieke Universiteit Leuven, Heverlee 3001, Belgium
{yuyi.wang,jan.ramon}@cs.kuleuven.be

Abstract. Graph support measures are functions measuring how frequently a given subgraph pattern occurs in a given database graph. An important class of support measures relies on overlap graphs. A major advantage of the overlap graph based approaches is that they combine anti-monotonicity with counting occurrences of a pattern which are independent according to certain criteria. However, existing overlap graph based support measures are expensive to compute.

In this paper, we propose a new support measure which is based on a new notion of independence. We show that our measure is the solution to a linear program which is usually sparse, and using interior point methods can be computed efficiently. We show experimentally that for large networks, in contrast to earlier overlap graph based proposals, pattern mining based on our support measure is feasible.

Keywords: Graph mining, frequent subgraph pattern mining, support measure, frequency counting, overlap graph, linear program.

1 Introduction

Graph mining is a subfield of structured data mining. An important task is *frequent subgraph pattern mining*, which concerns the problem of finding subgraph patterns that occur frequently in a collection of graphs or in a single large graph. In this paper, we consider the single-graph setting, and we will call the large graph containing all data the *database graph*. Referring to many applications, such as social networks, the Internet, chemical and biological interaction networks, traffic networks and citation networks, the database graph is also often called the *network*.

In order to define a frequent pattern mining problem precisely, a *support measure* (also called *frequency measure*) is needed. In the problem setting where patterns are mined in a set of transactions (e.g., itemset mining [1]), a simple support measure is to count the number of transactions in which the pattern occurs. However, in the context of a single large graph, the issue is less straightforward and several articles have considered this issue [2,4,5,6].

An important drawback of the strategy to just use the number of occurrences of a pattern (either embeddings or images) as its support is that it is not *anti-monotonic*, i.e., the support of a pattern may be larger than the support of

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 362–377, 2012.

one of its subpatterns. The anti-monotonicity of the support measure (or more generally interestingness measure) plays a very important role in the design of a pattern miner, as it allows for pruning the search space [7]. Nevertheless, anti-monotonicity alone is not enough. For example, a support measure just returning a constant is anti-monotonic, but not informative. From a statistical point of view, the value of a set of examples increases if these examples are more independent. Calders et al. [6] proposed to use the situation where occurrences of a subgraph pattern are independent (i.e., they do not overlap according to some notion of overlap) as a reference. In particular, the notion of a normalized graph support measure was defined: a support measure is *normalized* if for every pattern which has only non-overlapping occurrences in a database graph, its support in that database graph equals the number of occurrences.

An important class of support measures relies on *overlap graphs*. The vertices in an overlap graph represent occurrences of a given pattern, and two vertices are adjacent iff the corresponding occurrences overlap in the database graph (according to some notion of overlap, such as sharing a vertex or an edge). An overlap graph therefore summarizes how many times a pattern occurs in the database graph, and how independent these occurrences are. An overlap graph based support measure (OGSM) takes an overlap graph of a pattern in a database graph as its input, and outputs the support of that pattern in that database graph. Vanetik et al. [2] proposed the MIS measure, the size of the maximum independent set of the overlap graph. This is intuitively appealing since it measures how often we observed a pattern occurring independently. Unfortunately, computing the MIS of an overlap graph is NP-hard [8], and remains so even for bounded degree graphs. Moreover, it has been shown that MIS cannot be approximated even within a factor of $n^{1-o(1)}$ in polynomial time unless P=NP [9], where n is the order of the overlap graph. Calders et al. [6] proposed the Lovász theta function ϑ (see e.g., [10,11]), which is computable in time polynomial in the order of the overlap graph using semidefinite programming (SDP). A straightforward application of a general purpose SDP solver yields a running time of $O(n^{6.5})$ [17]. An SDP primal-dual algorithm for approximating ϑ with a multiplicative error of $(1+\epsilon)$ was proposed [12], and the running time of this algorithm is $O(\epsilon^{-2}n^5 \log n)$. Iyrngar et al. [15] considered subgradient methods for approximating ϑ, which run in time $O(\epsilon^{-2}\log^3(\epsilon^{-1})n^4 \log n)$ in the worst case. Unfortunately, even these approximative methods are still computationally too expensive for our purposes.

In this paper, we propose a new support measure `s` that is based on bounding the value of all occurrences of a pattern that share a particular part of the database graph, and `s` can be computed efficiently using a linear program (LP). The measure `s` is not a traditional OGSM, because its output does not depend only on the overlap graph considered in earlier papers. We introduce the notion *overlap hypergraph*, and `s` is an overlap hypergraph based support measure (OHSM). We prove that `s` is anti-monotonic and normalized. Furthermore, we show that all normalized anti-monotonic OHSMs are bounded. Our empirical analysis shows that this idea yields the first support measure which is

both overlap based (and hence appealing from a statistical point of view) and computationally feasible.

The remainder of this paper is structured as follows. In the next section, we briefly review some basic notation from graph theory and formalize support measures, overlap graphs and overlap hypergraphs. In Section 3, we introduce the new measure s and model it as an LP. We prove that s is normalized and anti-monotonic in Section 4. The property that all normalized anti-monotonic OHSMs are bounded is shown in Section 5. Section 6 points out a phase transition phenomenon between frequent and infrequent patterns. Section 7 presents experimental results. Section 8 concludes the paper with an overview of our contributions.

2 Preliminaries

2.1 Graph Theory

We recall basic graph theoretic notions used in this paper. For more background in this area, see also [13].

Graphs. A *graph* G is an ordered pair (V, E), where V is a set of *vertices* and E is either a set of *edges* $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ or a set of *arcs* $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$. In the former (latter) case, we call the graph *undirected* (*directed*). Vertices are *adjacent* if there is an edge (arc) between them. For an edge $e = \{u, v\}$ (arc $e = (u, v)$), u and v are *incident* with e.

A *labeled graph* is a quadruple $G = (V, E, \Sigma, \lambda)$, with (V, E) a graph, Σ a non-empty finite set of labels, and λ a function assigning labels in Σ to the vertices or edges (or arcs), or both. For simplicity, by labeled graph, we will mean *vertex-labeled graph* unless explicitly pointed out.

We will use the notation $V(G)$, $E(G)$ and λ_G to refer to the set of vertices, the set of edges (or arcs) and the labeling function of a graph G, respectively. g is said to be a *subgraph* of G if $V(g) \subseteq V(G)$, $E(g) \subseteq E(G)$ and for all $v \in V(g)$ that $\lambda_g(v) = \lambda_G(v)$, and write $g \subseteq G$.

We denote $\mathcal{G}$ the class of all graphs, and $\mathcal{G}^{\leftrightarrow}$ ($\mathcal{G}^{\rightarrow}$), the restriction to undirected (directed) graphs, while $\mathcal{G}_\lambda$ ($\mathcal{G}_\bullet$) denotes the restriction to labeled (unlabeled) graphs. One can combine notations, e.g., $\mathcal{G}_\bullet^{\rightarrow}$ for the class of directed, unlabeled graphs.

An *independent set* I of $G \in \mathcal{G}$ is a subset of $V(G)$ such that no pair of distinct vertices of I is adjacent in G. A *clique* Q of $G \in \mathcal{G}$ is a subset of $V(G)$ such that for all distinct vertices $v, w \in Q$, u and v are adjacent in G. A *clique partition* $\Pi = \{s_1, s_2, \cdots, s_k\}$ of $G \in \mathcal{G}$ is a partition of $V(G)$ such that every set s in Π is a clique.

Morphisms. The following concepts defined in terms of $\mathcal{G}_\lambda^{\rightarrow}$ are also valid for undirected and/or unlabeled graphs by dropping the direction of the edges and/or the labels of the vertices.

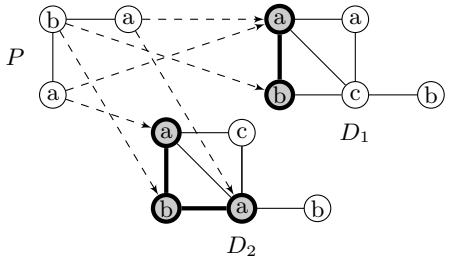

Fig. 1. Homomorphism and isomorphism. A homo-image (but not iso-image) of P is highlighted in D_1, and an iso-image of P is highlighted in D_2.

A *homomorphism* ψ from $G \in \mathcal{G}_\lambda^\rightarrow$ to $G' \in \mathcal{G}_\lambda^\rightarrow$ is a mapping from $V(G)$ to $V(G')$ such that for all $v \in V(G) : \lambda_G(v) = \lambda_{G'}(\psi(v))$ and for all $(u, v) \in E(G) : (\psi(u), \psi(v)) \in E(G')$. We call ψ *vertex-surjective* if $\forall v' \in V(G') : \exists v \in V(G) : \psi(v) = v'$, and call it *edge-surjective* if $\forall (u', v') \in E(G') : \exists (u, v) \in E(G) : \psi(u) = u'$ and $\psi(v) = v'$. A homomorphism is *surjective* if it is both vertex- and edge-surjective.

An *isomorphism* from $G \in \mathcal{G}_\lambda^\rightarrow$ to $G' \in \mathcal{G}_\lambda^\rightarrow$ is a bijective homomorphism ψ from G to G'. In this case, we say that G is *isomorphic* to G' and write $G \cong G'$. We use $G \subseteq G'$ to denote that $G \cong g$, for some subgraph g of G'. This is equivalent to saying that there exists a *subgraph isomorphism* from G to G'.

An *iso-image* (*homo-image*) g of $P \in \mathcal{G}_\lambda^\rightarrow$ in $D \in \mathcal{G}_\lambda^\rightarrow$ is a subgraph $g \subseteq D$ for which there exists an isomorphism (surjective homomorphism) ψ from P to g. We call g the iso-image (homo-image) through ψ. An individual isomorphism (homomorphism) ψ from P to g is called an *iso-embedding* (*homo-embedding*) of P in D. See Fig. 1 for an example.

In this paper, we only consider iso-images, although the measure `s` can be generalized for other matching operators such as homomorphism. We use the term *image* instead of iso-image afterwards, and denote with $\mathrm{Img}(D, P)$ the set of all images of P in D. Suppose $g \in \mathrm{Img}(D, p)$ and $g' \in \mathrm{Img}(D, P)$, if g is a subgraph of g', we call g a *subimage* of g' and g' a *superimage* of g.

Hypergraphs. A *hypergraph* is an ordered pair (V, E), where V is a set of vertices and E is a set of *hyperedges* $E \subseteq 2^V$. We denote $\mathcal{H}$ the class of all hypergraphs. As in the case of graphs, for $H \in \mathcal{H}$, $V(H)$ denotes the set of vertices and $E(H)$ denotes the set of hyperedges. To every hypergraph H which has n vertices and m hyperedges, we associate an $n \times m$ *incidence matrix* $M_H = (m_{ij})$ where $m_{ij} = 1$ if $v_i \in e_j$ and $m_{ij} = 0$ otherwise.

2.2 Support Measures

We review the concepts and properties of support measures and overlap graphs, and introduce the new concept of overlap hypergraphs.

Definition 1. *A* support measure *is a function* $f : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R}$ *that maps* (D, P) *to a non-negative number* $f(D, P)$*, where* P *is called the pattern,* D *the database graph and* $f(D, P)$ *the* support *of* P *in* D*.*

For efficiency reasons, most graph miners generate patterns from smaller patterns to larger ones [14]. Such a method requires the support measure to be anti-monotonic.

Definition 2. *A support measure* f *is* anti-monotonic *if for all* p, P, D *in* $\mathcal{G}$: $p \subseteq P \Rightarrow f(D, P) \leq f(D, p)$*.*

As explained in the introduction, anti-monotonicity alone is not enough. It is also desirable that the support measure accounts for the independence of the occurrences of the patterns. We can define *overlap* in different ways [6]. Popular definitions are *vertex-overlap*, i.e., two images g_1 and g_2 overlap if $V(g_1) \cap V(g_2) \neq \emptyset$, and *edge-overlap*, i.e., two images g_1 and g_2 overlap if $E(g_1) \cap E(g_2) \neq \emptyset$. Edge-overlap implies vertex-overlap. In this paper, by overlap, we will mean vertex-overlap, although our results are also valid in the edge-overlap setting.

Definition 3. *A support measure* f *is* normalized *if for all* P, D *in* $\mathcal{G}$: $f(D, P) = |Img(D, P)|$ *when there do not exist two distinct images* g_1 *and* g_2 *in* $Img(D, P)$ *satisfying* $V(g_1) \cap V(g_2) \neq \emptyset$*.*

Overlap Graphs. The notion of overlap graph plays an important role in the design and computation of anti-monotonic measures. Given a pattern P and a database graph D, the overlap graph of P in D is a graph $G_P^D \in \mathcal{G}_\bullet^\leftrightarrow$. Every vertex of G_P^D is an image of P in D, that is, $V(G_P^D) = \text{Img}(D, P)$. Two vertices u and v are adjacent in G_P^D if they overlap.

Vanetik et al. [3] define the induced support measure $f(G_P^D) = f(D, P)$, which we call an overlap graph based support measure (OGSM). They proposed the first normalized anti-monotonic OGSM, the size of the maximum independent set (MIS) [2]. Later, Calders et al. [6] proposed two normalized anti-monotonic OGSMs, the size of a minimum clique partition (MCP) and the Lovász theta value (ϑ). As mentioned in the introduction, these existing OGSMs are very expensive to compute.

Overlap Hypergraphs. As we are using vertex-overlap, each vertex v in a database graph D determines a clique in the overlap graph G_P^D in which P is a pattern. That is, suppose v is a vertex in D, then $Img_v(D, P) = \{g \in \text{Img}(D, P) \mid v \in V(g)\}$ build a clique in G_P^D since the images overlap at the vertex v.

We define the *overlap hypergraph* of P in D, denoted H_P^D as the hypergraph whose vertices are the images $\text{Img}(D, P)$, and for each vertex $v \in V(D)$ a hyperedge $e_v \in E(H_P^D)$ such that $e_v = \{g \in V(H_P^D) \mid v \in V(g)\}$. The hyperedges represent cliques in G_P^D.

In an overlap hypergraph H_P^D, we say that a hyperedge e is *dominated* by another hyperedge e' if $e \subset e'$, and a hyperedge e is *dominating* if it is not

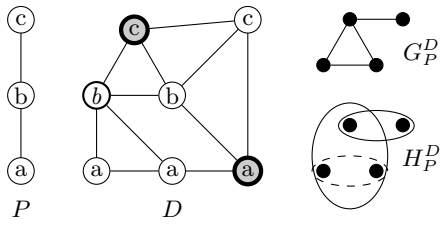

Fig. 2. Overlap graph and overlap hypergraph. Given a pattern P, a database graph D, the overlap graph G_P^D and the overlap hypergraph H_P^D are shown on the right. In the overlap hypergraph, the (dominating) hyperedges are determined by the highlighted vertices in the database graph, and a dominated hyperedge is given in a dashed ellipse.

dominated by any other hyperedge. For any D and P, we define the reduced overlap hypergraph $\tilde{H}_P^D$ to be the hypergraph for which $V(\tilde{H}_P^D) = V(H_P^D)$ and $E(\tilde{H}_P^D)$ is the set of all dominating hyperedges of H_P^D. In the sequel we only refer to $\tilde{H}_P^D$. We will abuse terminology and simply call $\tilde{H}_P^D$ the overlap hypergraph. See Fig. 2 for an example.

We henceforth refer to the induced support measure, which we denote by $f(\tilde{H}_P^D)$, instead of referring to $f(D, P)$. Such induced support measures are called overlap hypergraph based support measures (OHSM). We call OHSMs and OGSMs overlap based support measures.

3 A New Normalized Anti-monotonic Measure

We introduce a new normalized anti-monotonic OHSM, which we denote s. It satisfies the desirable properties of being anti-monotic and normalized, and can be computed efficiently.

The MIS measure is a normalized anti-monotonic OGSM. Note that given an overlap hypergraph $\tilde{H}_P^D$, we are able to derive the corresponding overlap graph G_P^D by replacing every hyperedge with a clique. Therefore, we can rephrase the definition of the MIS measure using overlap hypergraphs. Suppose $\tilde{H}_P^D$ is an overlap hypergraph:

$$MIS(\tilde{H}_P^D) = \max |\{I \subseteq V(\tilde{H}_P^D) \mid \forall e \in E(\tilde{H}_P^D) : |e \cap I| \leq 1\}| \tag{1}$$

The MIS measure requires that a vertex of an overlap (hyper)graph is either in the independent set I or not. Our new measure s is a relaxation of the MIS measure by allowing counting vertices of an overlap hypergraph partially.

Let $\tilde{H}_P^D$ be an overlap hypergraph. We start by assigning to each vertex v of $\tilde{H}_P^D$ a variable x_v. We then consider vectors $x \in \mathbb{R}^{V(\tilde{H}_P^D)}$ of variables where for every $v \in V(\tilde{H}_P^D)$, x_v denotes the variable (component of x) corresponding to v. x is *feasible* iff it satisfies

$$\text{(i) } \forall v \in V(\tilde{H}_P^D) : 0 \leq x_v$$
$$\text{(ii) } \forall e \in E(\tilde{H}_P^D) : \textstyle\sum_{v \in e} x_v \leq 1.$$

We denote the feasible region (the set of all feasible $x \in \mathbb{R}^{V(\tilde{H}_P^D)}$) with $\mathfrak{R}(\tilde{H}_P^D)$. It is a convex polytope. The measure s is defined by

$$\mathsf{s}(\tilde{H}_P^D) = \max_{x \in \mathfrak{R}(\tilde{H}_P^D)} \sum_{v \in V(\tilde{H}_P^D)} x_v \tag{2}$$

Clearly, s is the solution to a linear program.

We will call an element $x \in \mathfrak{R}(\tilde{H}_P^D)$ which makes $\sum_{v \in V(\tilde{H}_P^D)} x_v$ maximal a *solution* to the LP of s.

There are very effective methods for solving LPs, including the simplex method which is efficient in practice though its complexity is exponential, and the more recent interior-point methods [16]. The interior-point method solves an LP in $O(n^2 m)$ time, where n (here $\min\{|V(\tilde{H}_P^D)|, |E(\tilde{H}_P^D)|\}$) is the number of variables, and m (here $|V(\tilde{H}_P^D)| + |E(\tilde{H}_P^D)|$) is the number of constraints. Usually, patterns are not large, so the LPs for computing s are sparse. Almost all LP solvers perform significantly better for sparse LPs.

4 Conditions for Anti-monotonicity

Vanetik et al. [3] gave necessary and sufficient conditions for anti-monotonicity of for OGSMs on labeled graph using edge-overlap. This result was generalized in [6] to any OGSM on labeled or unlabeled, directed or undirected graphs using edge overlap or vertex overlap and isomorphism, homomorphism or homeomorphism. Our conditions for anti-monotonicity are based on the overlap hypergraphs. Our main result is that an OHSM is anti-monotonic if and only if it is non-decreasing under certain operations on the overlap hypergraph.

We begin by defining three operations on any overlap hypergraph, which we will then use in our conditions for anti-monotonicity. These operations are different from those used in [3,6], but play a similar role. As mentioned in these earlier papers, the motivation for these operations is that it is often easier to show that an OHSM satisfies the conditions of the theorem (being non-decreasing under the three operation), than to show anti-monotonicity of a measure directly.

For $H \in \mathcal{H}$, we define:

- Vertex Addition: A new vertex v is added to every existing hyperedge: $VA(H, v) = (V(H) \cup \{v\}, \{e \cup \{v\} \mid e \in E(H)\})$.
- Subset Contraction: Let $K \subseteq V(H)$ be a set of vertices of the hypergraph such that $\exists e \in E(H) : K \subseteq e$. Then, the subset contraction operation contracts K into a single vertex k, which remains in only those hyperedges that are supersets of K. Formally, $SC(H, K, k) = (V(H) - K \cup \{k\}, E_1 \cup E_2)$ where $E_1 = \{e - K \cup \{k\} \mid e \in E(H) \text{ and } K \subseteq e\}$ and $E_2 = \{e - K \mid e \in E(H) \text{ and } K \nsubseteq e\}$).
- Hyperedge Split: This operation splits a size k hyperedge into k hyperedges of size $(k-1)$ each: $HS(H, e) = (V(H), E(H) - \{e\} \cup \{e - \{v\} \mid v \in e\})$, where $e \in E(H)$.

For example, suppose H_0 is a hypergraph, $V(H_0) = \{v_1, v_2, v_3, v_4\}$, and $E(H_0)$ contains two hyperedges $\{v_1, v_2, v_3\}$ and $\{v_1, v_4\}$. Let $H_1 = VA(H_0, v_5)$, then $V(H_1) = \{v_1, v_2, v_3, v_4, v_5\}$ and $E(H_1)$ contains hyperedges $\{v_1, v_2, v_3, v_5\}$ and $\{v_1, v_4, v_5\}$. Let $H_2 = SC(H_1, \{v_1, v_3\}, v_6)$, then $V(H_2) = \{v_2, v_4, v_5, v_6\}$ and $E(H_2)$ contains hyperedges $\{v_2, v_5, v_6\}$ and $\{v_4, v_5\}$. Let $H_3 = HS(H_2, \{v_2, v_5, v_6\})$, then $V(H_3) = V(H_2)$ and $E(H_2)$ contains four hyperedges $\{v_2, v_5\}$,$\{v_2, v_6\}$, $\{v_5, v_6\}$ and $\{v_4, v_5\}$.

4.1 Sufficient Condition

We give a sufficient condition for support measure anti-monotonicity in terms of the three operations on the overlap hypergraph that we have defined.

Theorem 1. *Let $f' : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ be a support measure, and $f : \mathcal{H} \to \mathbb{R}$ with $f'(D, P) = f(\tilde{H}_P^D))$ be the induced OHSM. If f is non-decreasing under VA, SC and HS, then f' is an anti-monotonic support measure.*

Proof. Suppose D is a database graph, and p and P are two patterns such that p is a subgraph of P. We prove that $\tilde{H}_p^D$ can be obtained from $\tilde{H}_P^D$ by applying only the operations VA, SC and HS. It follows then that $f'(D, P) = f(\tilde{H}_P^D) \leq f(\tilde{H}_p^D) = f'(D, p)$ for any D, P and p, proving the theorem.

Let $<$ be an arbitrary order defined on $V(\tilde{H}_p^D)$. We define for $v \in V(\tilde{H}_p^D)$ the set $\Pi_v = \{u \in V(\tilde{H}_P^D) \mid v \preceq u \text{ and } \forall w < v : w \not\preceq u\}$. Here, remember that the vertices of $\tilde{H}_p^D$ are images of p and hence $v \preceq u$ refers to a subgraph isomorphism relationship between v and u.

The Π_v are pairwise disjoint and $\cup_{v \in V(\tilde{H}_p^D)} \Pi_v = V(\tilde{H}_P^D)$. We point out that there may exist vertices v for which $\Pi_v = \emptyset$. We divide $V(\tilde{H}_p^D)$ into two sets $V_0 = \{v \mid \Pi_v = \emptyset\}$ and $V_1 = \{v \mid \Pi_v \neq \emptyset\}$.

Let H be a hypergraph initially equal to $\tilde{H}_P^D$. We will perform operations VA, SC and HS on H, until finally it is equal to $\tilde{H}_p^D$.

First, H is modified by a sequence of VA operations. For each $v \in V_0$, we do $H := VA(H, v)$. Now, $\forall e \in E : V_0 \subseteq e$.

Then, for each $v \in V_1$, we perform $H := SC(H, \Pi_v, v)$. The operations are valid because for $v \in V_1$ each vertex $u \in \Pi_v$ stands for a superimage of the same v, i.e., $v \preceq u$ and hence $\exists e \in E(H) : \Pi_v \subseteq e$. It is easy to verify that now $V(\tilde{H}_p^D) = V(H)$ holds.

Consider a hyperedge $e'_x \in E(\tilde{H}_p^D)$ which is determined by $x \in V(D)$, i.e., $e'_x = \{v \in V(\tilde{H}_p^D) \mid x \in V(v)\}$. We know that $e'_x \cap V_0$ is a subset of any $e \in E(H)$. $E(\tilde{H}_P^D)$ has a dominating hyperedge e''_x determined by x, i.e., $e''_x = \{v \in V(\tilde{H}_P^D) \mid x \in V(v)\}$ (or has another hyperedge e''_y which is a superset of the dominated hyperedge e''_x). We have $e'_x \subseteq e''_x$ (or e''_y). Thus, $\forall v \in e'_x \cap V_1 : \Pi_v \subseteq e''_x$ (or e''_y). Therefore, there must be a hyperedge $e \in E(H)$ such that $e'_x \subseteq e$. This property shows that every hyperedge in $E(\tilde{H}_p^D)$ either exists in $E(H)$ or can be obtained later on by performing a sequence of HS on H. □

Theorem 2. $\mathsf{s}(D, P) = \mathsf{s}(\tilde{H}_P^D)$ *is a normalized anti-monotonic support measure.*

Proof. First, we prove s is normalized. If the pattern P only has non-overlapping images in the database graph D, every hyperedge in $E(\tilde{H}_P^D)$ contains only one vertex, then setting $x_v = 1$ for every $v \in V(\tilde{H}_P^D)$ is a feasible assignment and is clearly maximal. That is, s equals the number of non-overlapping images. Therefore, s is normalized.

Then, we prove s is anti-monotonic using Theorem 4.1. Suppose H is an overlap hypergraph and x^* is a solution to the LP of $\mathsf{s}(H)$. Let H_1 be the overlap hypergraph $VA(H, v)$, and let $x_u = x_u^*$ for all vertices $u \neq v$ and $x_v = 0$. x is a feasible solution for the LP of $\mathsf{s}(H_1)$, so $\mathsf{s}(H_1) \geq \sum_v x_v = \mathsf{s}(H)$. Let H_2 be the overlap hypergraph $SC(H, K, k)$, and let $x_u = x_u^*$ for all vertices $u \neq k$ and $x_k = \sum_{v \in K} x_v^*$. x is a feasible for the LP of $\mathsf{s}(H_2)$, so $\mathsf{s}(H_2) \geq \sum_v x_v = \mathsf{s}(H)$. Let H_3 be the overlap hypergraph $HS(H, e)$. x^* is also a feasible for the LP of $\mathsf{s}(H_3)$, so $\mathsf{s}(H_3) \geq \mathsf{s}(H)$. □

4.2 Necessary Condition

We show that the above sufficient condition for anti-monotonicity is also necessary.

Theorem 3. *Let $f' : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ be a support measure, and $f : \mathcal{H} \to \mathbb{R}$ with $f'(D, P) = f(\tilde{H}_P^D))$ be the induced OHSM. If f' is anti-monotonic, then f is non-decreasing under VA, SC and HS.*

Proof (sketch). Let H_P be any hypergraph and H_p a hypergraph obtained by performing VA, SC or HS on H_P. We show that there exists a database graph D and patterns P and p such that $\tilde{H}_P^D = H_P$ and $\tilde{H}_p^D = H_p$. Then, there follows $f(H_P) = f'(D, P) \leq f'(D, p) = f(H_p)$ which proves the theorem. For convenience, we show the theorem only for $D, P, p \in \mathcal{G}_\lambda^{\leftrightarrow}$, but the proof can be generalized.

In Figure 3, we give the patterns P and p ($p \subseteq P$), and list different types of overlap. The numbers of vertices with label a or b in P and p are not fixed, and we can assume that P and p have enough such vertices. We construct database graphs by combining the patterns using these different types of overlap. We name the different types O_1, O_2, O_3 and O_4. In Figure 3, only two patterns overlap for each type, but during the construction of database graphs, it is allowed that more than two patterns overlap at the same vertex.

If $\tilde{H}_p^D = VA(\tilde{H}_P^D, v)$, then we can construct the database graph using O_1 and O_2. O_1 is used to determine all the hyperedges in $E(\tilde{H}_P^D)$. O_2 is used to introduce a new vertex and make the new vertex exist in every hyperedge in $E(\tilde{H}_p^D)$.

If $\tilde{H}_p^D = SC(\tilde{H}_P^D, K, k)$, then we can construct the database graph using O_1, O_3 and O_4. O_3 is used to build the subset K. O_4 is used to determine the hyperedges $e \in E(\tilde{H}_P^D)$ which satisfy $e \cap K \neq \emptyset$ and $K \nsubseteq e$. For any hyperedge e determined by O_4, there is a hyperedge $e' \in E(\tilde{H}_p^D)$ determined by O_1 such that $e' = e - K$. Besides, O_1 also determines all hyperedges $e \in \tilde{H}_P^D$ which satisfy $K \subseteq e$ or $e \cap K = \emptyset$.

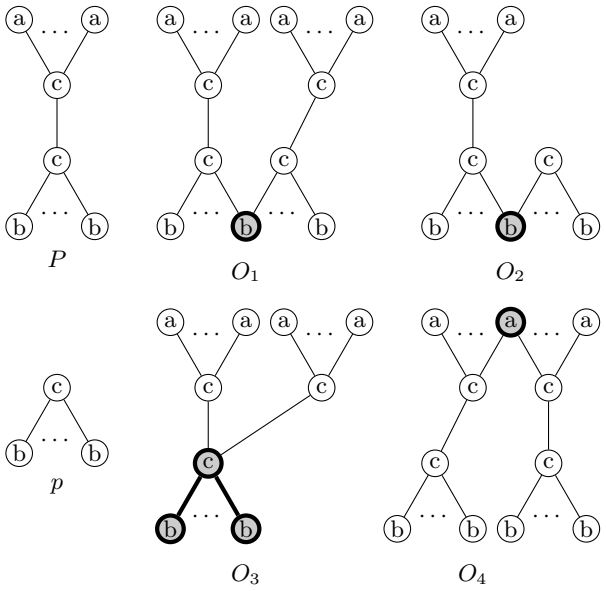

Fig. 3. Patterns and different types of overlap. The highlighted parts show the ways two patterns overlap.

If $\tilde{H}_p^D = HS(\tilde{H}_P^D, e)$, then we can construct the database graph using O_1 and O_4. O_4 is used to build the hyperedge e. O_1 determines the hyperedges $\{e - \{v\} \mid vn \in e\} \in E(\tilde{H}_p^D)$ and all other hyperedges. □

5 Bounding Theorem

In [6], the authors showed an interesting result that all normalized anti-monotonic OGSMs are bounded (between the maximum independent set size (MIS) and the minimum clique partition size (MCP)). Similarly, we prove that all normalized anti-monotonic OHSMs are also bounded. We first introduce another OHSM on $H \in \mathcal{H}$, the size of a minimum set cover of H:

$$MSC(H) = \min |\{S \subseteq E(H) \mid \bigcup_{e \in S} e = V(H)\}| \tag{3}$$

It is not difficult to verify that MSC is normalized and anti-monotonic. To compute MSC is an NP-hard problem. The maximum independent set size (Eq. (1)) and minimum vertex cover (Eq. (3)) are the minimal and the maximal possible normalized anti-monotonic OHSMs.

Theorem 4. *For every normalized anti-monotonic OHSM f, and every $H \in \mathcal{H}$, it holds that: $MIS(H) \leq f(H) \leq MSC(H)$.*

Proof. We use Theorem 3 to show the minimality of MIS and the maximality of MCP, respectively.

Let H be a hypergraph, and let $I = \{v_1, v_2, \cdots, v_k\}$ be a maximum independent set of H. Starting from the hypergraph $H_I = (\{v_1, v_2, \cdots, v_k\}, \{\{v_1\}, \{v_2\}, \cdots, \{v_k\}\})$, we can get H by adding vertices $V(H) - I$ using VA first and then spliting hyperedges by a sequence of HS. Since f is normalized, it is anti-monotonic and therefore f cannot decrease after each step, and $f(H_I) = k$. As such, $f(H)$ is larger than or equal to $k = MIS(H)$.

On the other hand, let $\{e_1, e_2, \cdots, e_k\}$ be a minimum set cover for H and let $H_{sc} = SC(\ldots SC(SC(H, e_1, v_{e_1}), e_2, v_{e_2}) \cdots, e_k, v_{e_k})$. H_{sc} only has the hyperedges with exact one vertex in each of them. Because f is anti-monotonic, f is not decreasing under SC and thus $f(H) \leq f(SC(H, e_1)) \leq \cdots \leq f(H_{sc}) = k$.

□

6 The Phase Transition from Frequent to Infrequent

Large real-world networks are known to satisfy properties similar to random graphs. A well-known property is that properties which can be expressed in first order logic are satisfied by either almost all graphs or almost no graphs (0-1 law, see [21]). For random graphs, one can observe (see also our experiments below) that for a given pattern P, it is either very easy to embed the pattern in the network, or very difficult. This leads to another 0-1 property: the frequency of many patterns is either very low or very high (for our s measure, nearly equal to the network size). Consider e.g. a social network and the pattern "X is a friend of Y and Y is a friend of Z". Since most people have at least two friends, such pattern will match about everywhere. This holds more generally for many tree and path patterns. In fact, most such patterns are overly general and not very interesting.

In the context of overlap-graph based support measures, these overly general patterns also pose a computational problem: since they match about everywhere, the corresponding overlap graph is very large. Therefore, for these less interesting overly general patterns, our prototype implementation just records that they are very frequent but doesn't attempt to compute their frequency exactly by constructing the overlap graph explicitely. We hence distinct three categories of patterns: the infrequent patterns, the moderately frequent patterns, and the very frequent patterns (for which the frequency will not be computed exactly).

7 Experiments

This section provides experimental results, illustrating the practical potential of our new measure s.

7.1 Experimental Setup

For our experiments, we are interested in answering the following experimental questions:

Q1 How does the computational cost of the s measure compare to other existing overlap based support measures, e.g., Lovász ϑ value?
Q2 How does the cost of computing the s measure compare to the cost of listing the embeddings?
Q3 Is it feasible to mine all s-frequent patterns of size up to 6 in moderately sized networks?
Q4 What can we learn about the phase transition between frequent and infrequent and the randomness of the DBLP dataset?

7.2 Results

All experiments are run on an Intel Core i7-2600 CPU (3.4Gz) with 8Gb RAM.

We use the algorithm VF2 (implemented in C++) to find embeddings of patterns in networks [19]. We use Matlab 2012*a* and SeduMi 1.21 to solve the LPs for the s measure and the SDPs for the Lovász ϑ value. The desired accuracy of all LPs and SDPs is 10^{-4}.

Lovász ϑ Function. In the first experiment, we generate hypergraphs randomly, and convert them into graphs by replacing the hyperedges with cliques. The hypergraphs are used to compute s measures, while the graphs are used to compute the Lovász ϑ measure. The hypergraphs have $20, 40, \ldots, 200$ vertices and $20, 40, \ldots, 100$ hyperedges. With probability 0.05, a vertex of the hypergraphs appears in a hyperedge.

Fig. 4 shows the time cost to compute the s measure and the Lovász ϑ measure for these graphs. θ_m and s_m means there are m hyperedges.

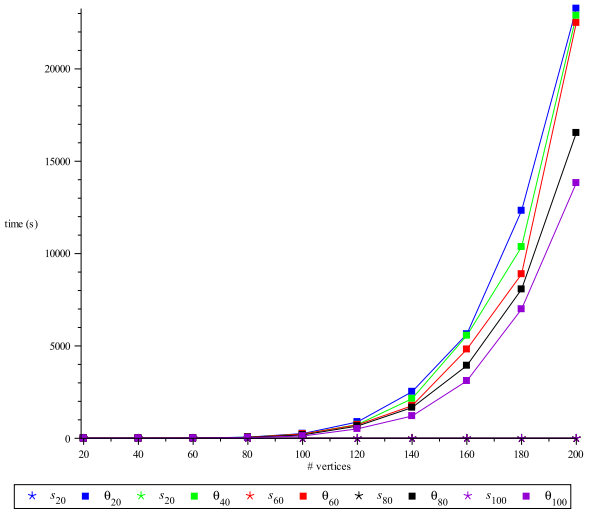

Fig. 4. Time consumed to compute θ and s

Real-World Data. We use two DBLP co-authorship networks (DBLP0305 showing co-authorships from 2003 to 2005 and DBLP0507 showing co-authorships from 2005 to 2007) [18]. If an author i co-authored a paper with author j, the networks contain an undirected edge $\{i, j\}$. The vertices are unlabeled, whereas the edges are labeled with an integer indicating the year the edge first appeared in. The network dblp0305 has 109944 vertices, 228461 edges and 3 different labels. The network dblp0507 has 135516 vertices, 290363 edges and 3 different labels. In this experiment, we choose 1.5% as the frequency threshold.

For each network, we start from patterns of level 1, the single vertices. A pattern which has $i-1$ edges is a *candidate* in level i ($i \geq 2$) if none of its subpatterns is infrequent and at least one of its subpatterns in level $i-1$ is frequent (others may have too many embeddings). If a pattern has more than 5.10^6 embeddings, we don't compute s but can easily show that the pattern is frequent. We call such a pattern *very frequent*.

Table 1 gives the results of the experiments that mine frequent patterns up to level 6 in the DBLP networks. T_{map} is the average time per pattern to find embeddings using VF2, and T_{s} is the average time to compute s. Both are in seconds.

Synthetic Data. We generate scale-free networks of different sizes [20]. They have $10^2, 10^3, \ldots, 10^6$ vertices which are labeled by 4 different labels, and all of them have the same average degree 10. We will call them 10_X networks, where $X = 2, 3, 4, 5, 6$. In this experiment, all tree patterns are very frequent. Therefore, we only report statistics for the non-tree patterns. We choose the frequency threshold 0.1%.

Tables (2)-(4) give the results of the experiments that mining frequent non-tree patterns up to level 6 (except the network which has 10^6 vertices) in the

Table 1. Frequent pattern mining in DBLP0305 and DBLP0507. Lev. = level (pattern size), Cand. = # candidate patterns, Comp. = # patterns for which s was computed, Freq. = # frequent patterns

Lev.	Cand.	Comp.	Freq.	T_{map}	T_{s}	Cand.	Comp.	Freq.	T_{map}	T_{s}
1	1	1	1	0.452	0.000251	1	1	1	0.711	0.000303
2	3	3	3	10.783	2.041	3	3	3	11.225	2.743
3	6	6	6	24.166	8.022	6	6	6	37.152	22.245
4	34	28	19	92.035	41.557	34	28	22	73.531	77.161
5	95	25	8	634.099	42.234	118	54	25	814.156	138.145
6	56	13	7	817.789	91.018	179	35	12	1530.608	430.637

Table 2. Frequent non-tree pattern mining in the 10_2 network

Level	Candidates	Computed	Frequent	T_{map}	T_{s}
4	20	20	16	0.014	0.383
5	191	191	182	0.015	0.388
6	2083	2083	2033	0.018	0.394

Table 3. Frequent non-tree pattern mining in the 10_3 and 10_4 network

Lev.	Cand.	Comp.	Freq.	T_{map}	T_{s}	Cand.	Comp.	Freq.	T_{map}	T_{s}
4	20	20	20	0.018	0.383	20	20	20	0.085	0.393
5	215	215	215	0.031	0.431	215	215	215	0.277	0.406
6	2430	2430	2422	0.128	0.481	2430	2430	2349	3.141	1.877

Table 4. Frequent non-tree pattern mining in the 10_5 and 10_6 networks

Lev.	Cand.	Comp.	Freq.	T_{map}	T_{s}	Cand.	Comp.	Freq.	T_{map}	T_{s}
4	20	20	5	2.906	0.301	20	20	0	216.196	0.428
5	99	99	9	12.435	0.673	55	55	12	1565.134	0.996
6	758	742	648	354.194	24.408	-	-	-	-	-

scale-free networks. Levels 1 to 3 only contain tree patterns, so we do not list them in the tables.

7.3 Discussion

Based on the results presented above, we can answer the experimental questions as follows:

Q1 One can see from Table 4 that, for all the randomly generated (hyper)graphs, s can be computed in a very short period of time (< 0.01 seconds), while the time consumed to compute ϑ grows fast when the number of vertices increases. Clearly, for larger (hyper)graphs on which s measure can be computed efficientlly, it is extremely difficult to compute the ϑ value in a reasonable time period by solving the corresponding SDP using existing methods. Therefore, s outperforms ϑ value in terms of efficiency.

Q2 On the real-world data, the time needed to compute embeddings is significantly larger than the time needed to compute s. For the larger synthetic datasets and the larger patterns the difference is even several orders of magnitude.

Q3 We can see that using VF2 and the s measure, frequent patterns of level up to 6 can be mined in a reasonable amount of time. In contrast to earlier approaches using the MIS or ϑ measures, here the bottleneck is clearly the pattern matching part of the algorithm. If this part can be improved, it can be expected that larger patterns can be mined in larger networks.

Q4 For the synthetic data, we found that the frequency of cyclic (non-tree) patterns was rather low, we needed a frequency threshold of 0.1% to mine them. One can conclude that while in standard random graph models nodes choose their neighbors randomly, in real-world data the connections of candidate neighbors have an important influence.

8 Conclusions

In this paper, we studied the problem of measuring how frequently a given pattern occurs in a given database graph. We have proposed a new overlap based

support measure s. In contrast to existing overlap based support measures, it can be computed efficiently. We have shown that it is anti-monotonic and normalized. The experimental results demonstrate that it is a practical overlap based measure and it is effective to prune the search space.

Compared to non-overlap based measures, e.g., the min-image support measure [4], the s measure has statistical advantages. For example, consider the embeddings: $\langle 1, 11\rangle, \langle 2, 11\rangle, \langle 3, 11\rangle, \langle 4, 11\rangle, \langle 5, 11\rangle, \langle 6, 12\rangle, \langle 6, 13\rangle, \langle 6, 14\rangle, \langle 6, 15\rangle$ and $\langle 6, 16\rangle$. Then min-image returns 6 while s returns 2. The latter equals the number of independent embeddings. Therefore, from a statistical point of view, for counting the number of independent observations of some phenomenon s is preferable.

This aim to measure only independent occurrences is shared with the MIS measure [3]. MIS is NP-hard while s is an efficiently computable relaxation. MIS returns an integer and is more strict in the sense that it never accounts for overlapping occurrences, while s also partially counts observations not explained by vertices of already counted embeddings. E.g. consider the embeddings $\langle a, b, c\rangle$, $\langle a, d, e\rangle$ and $\langle f, b, e\rangle$. The MIS is 1. However, even though each of the vertices a, b and e could have 'caused' two embeddings, no vertex is involved in all three embeddings. Therefore, s partially counts the third embedding, in this case resulting in the value 1.5.

Our proposed measure is flexible, in the sense that it is possible for a user to plug in his own definition of overlap. Investigating this in more detail is one possible line of future research. Our proposal makes measuring the frequency of a pattern in a more sound statistical way tractable. There are however other challenges related to pattern mining in networks. The major one in our experiments was the pattern matching. However, we anticipate that here too we can get a long way in making things tractable. In particular we intend to integrate our approach with recent results concerning efficient pattern matching operators based on arithmetic circuits [22].

Acknowledgements. This work was supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks: a Theory-based appraoch".

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of SIGMOD 1993, pp. 207–216 (1993)
2. Vanetik, N., Gudes, E., Shimony, S.E.: Computing frequent graph patterns from semistructured data. In: Proceeding of ICDM 2002, pp. 458–465 (2002)
3. Vanetik, N., Shimony, S.E., Gudes, E.: Support measures for graph data. Data Min. Knowl. Discov. 13(2), 243–260 (2006)
4. Bringmann, B., Nijssen, S.: What Is Frequent in a Single Graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
5. Fiedler, M., Borgelt, C.: Support Computation for Mining Frequent Subgraphs in a Single Graph. In: Proceedings of MLG 2007 (2007)

6. Calders, T., Ramon, J., Dyck, D.V.: All normalized anti-monotonic overlap graph measures are bounded. Data Min. Knowl. Discov. 23(3), 503–548 (2011)
7. Kuramochi, M., Karypis, G.: Finding frequent patterns in a large sparse graph. Data Min. Knowl. Discov. 11(3), 243–271 (2005)
8. Garey, M.R., Johnson, D.S.: Computers and intractibility, a guide to the theory of NP-Completeness. W. H. Freeman and Company (1979)
9. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Approximating clique is almost NP-Complete. In: FOCS, pp. 2–12. IEEE Computer Society (1991)
10. Lovász, L.: On the Shannon capacity of a graph. IEEE Transactions on Information Theory 25(1), 1–7 (1979)
11. Knuth, D.E.: The sandwich theorem. Electr. J. Comb. 1, 1–48 (1994)
12. Chan, T., Chang, K.L., Raman, R.: An SDP primal-dual algorithm for approximating the Lovsz-theta function. In: Proceedings of the IEEE ISIT 2009, pp. 2808–2812 (2009)
13. Diestel, R.: Graph theory. Springer (2010)
14. Chakrabarti, D., Faloutsos, C.: Graph mining: laws, generators, and algorithms. ACM Comput. Surv. 38(1), 1–69 (2006)
15. Iyengar, G., Phillips, D.J., Stein, C.: Approximating semidefinite packing programs. SIAM Journal on Optimization 21(1), 231–268 (2011)
16. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge Univ. Press (2004)
17. Klein, P.N., Lu, H.: Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In: Proc. of ACM STOC 1996, pp. 338–347 (1996)
18. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining Graph Evolution Rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
19. Luigi, P., Pasquale, F., Carlo, S., Mario, V.: A subgraph isomorphism algorithm for matching large graphs. IEEE Trans. Pat. Anal. Mach. Intell. 26(10), 1367–1372 (2004)
20. Barabási, A., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
21. Fagin, R.: Probabilities on finite models. J. of Symbolic Logic 41(1), 50–58 (1976)
22. Kibriya, A., Ramon, J.: Nearly exact mining of frequent trees in large networks. In: Proceedings of ECML-PKDD 2012 (in press)