GraphRank: Statistical Modeling and Mining of Significant Subgraphs in the Feature Space

Huahai He Ambuj K. Singh
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106, USA
{huahai, ambuj}@cs.ucsb.edu

Abstract

We propose a technique for evaluating the statistical significance of frequent subgraphs in a database. A graph is represented by a feature vector that is a histogram over a set of basis elements. The set of basis elements is chosen based on domain knowledge and consists generally of vertices, edges, or small graphs. A given subgraph is transformed to a feature vector and the significance of the subgraph is computed by considering the significance of occurrence of the corresponding vector. The probability of occurrence of the vector in a random vector is computed based on the prior probability of the basis elements. This is then used to obtain a probability distribution on the support of the vector in a database of random vectors. The statistical significance of the vector/subgraph is then defined as the p-value of its observed support. We develop efficient methods for computing p-values and lower bounds. A simplified model is further proposed to improve the efficiency. We also address the problem of feature vector mining, a generalization of itemset mining where counts are associated with items and the goal is to find significant sub-vectors. We present an algorithm that explores closed frequent sub-vectors to find significant ones. Experimental results show that the proposed techniques are effective, efficient, and useful for ranking frequent subgraphs by their statistical significance.

1 Introduction

Recent advances in science and technology have generated a large amount of complex data. As a powerful abstract data type, graphs are often used to represent these complex data. In the database community, graph models have been used for schema matching [1], web documents, multimedia [2], and social networks [3]. In biology, graphs have been used to represent molecular structures, protein 3D structures [4], and protein interaction networks [5].

Mining structured patterns in a collection of graphs is useful for understanding the intrinsic characteristics of scientific data. In drug development, frequent pattern mining can reveal conserved substructures in a category of medically effective chemical compounds [6]. In studies of protein interaction networks, conserved patterns in multiple species reveal cellular machinery [5]. In the analysis of protein structures, the presence of conserved subgraphs in protein contact maps can reveal evolutionarily significant patterns of chemical bonds and interactions [4].

A number of techniques have been developed to find frequent subgraphs [7, 8, 9, 10, 11, 12, 13, 14] in a transactional database, i.e., a large collection of graphs. However, the usefulness of frequent subgraph mining is limited by two factors:

1. Not all frequent subgraphs are *statistically significant*.
2. There is no way to *rank* the frequent subgraphs. This hinders the identification of subgraphs of real interest, especially when the number of discovered frequent subgraphs is large.

For illustrative purposes, consider a sample graph database shown in Figure 1 and some frequent subgraphs shown in Figure 2. The *support* of a subgraph is the number of graphs that contain the subgraph. A subgraph is *frequent* if its support is above a given threshold. Neither the support nor the size of a subgraph is sufficient to measure the statistical significance of a subgraph, and to rank the listed subgraphs.

1.1 Our Approach

In this paper, we propose a technique for computing the statistical significance of frequent subgraphs, and show that frequent subgraphs can be effectively ranked by this measure.

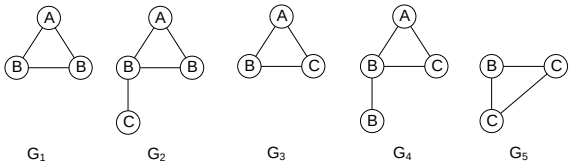


Figure 1. A sample graph database

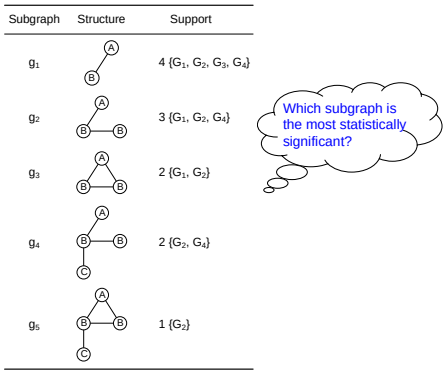


Figure 2. Frequent subgraphs and their supports

Given a subgraph g and its observed support μ_0, its statistical significance is defined as the probability that g occurs in a database of random graphs with a support $\mu \geq \mu_0$, namely the *p-value* of g. In this way, we can compute the p-values of all frequent subgraphs discovered by existing subgraph mining techniques, rank them by p-values, and/or remove insignificant ones. This would greatly improve the quality of the mining results.

The main challenge of the above procedure is how to estimate the probability that a subgraph occurs in a random graph. As graphs have flexible structures, it is difficult to estimate such probability directly in the graph space (Note that the problem of determining whether a graph is a subgraph of another is NP-complete). Milo et al [15] adopted a simulation approach: generate many random graphs while maintaining some empirical measures such as degree of vertices, number of edges, and then count the ones that contain the subgraph. However, this approach is neither scalable to a large collection of graphs nor precise for computing and comparing small p-values.

We address the above challenge by transforming graphs into a feature space. First, we use domain knowledge to define a set of basis elements such as vertices, edges, or small subgraphs. A graph is simply regarded as a collection or a histogram of basis elements; this defines its feature vector. Then, we approximate the question of significance of a subgraph by considering the significance of its feature vector in the feature space. This is a simpler problem that admits closed-form solutions. Although structural information of a graph is lost in the feature space, statistics on the basis elements are still captured. As shown by the experimental results, this approximation is suitable for the discovery of significant subgraphs.

Figures 3 and 4 outline our approach. In the first phase (Figure 3), we obtain frequent subgraphs from a target graph database using existing graph mining techniques, and transform them into feature vectors. In the second phase (Figures 4), we compute the probability that feature vector $\underline{x}$ of a subgraph g occurs in a random vector, and use this probability to compute the probability distribution on $\underline{x}$'s support in a random database. The statistical significance of g is then computed as the p-value of its observed support in the target database.

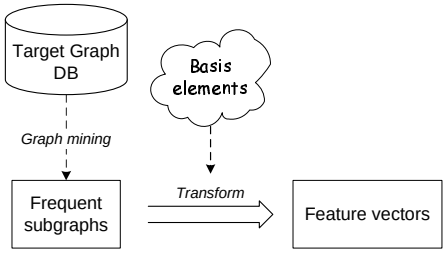


Figure 3. Representation of graphs as feature vectors

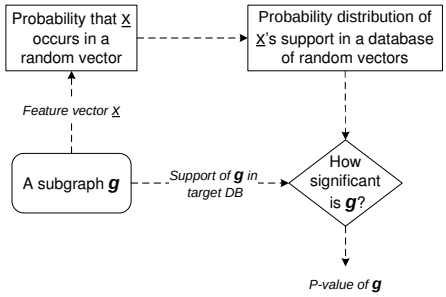


Figure 4. Computation of p-value of a frequent subgraph

In the second half of the paper, we address the problem of feature vector mining, a simplified version of graph mining. Vector (aka histogram and multiset) mining is an important generalization of frequent itemset mining. We develop ClosedVect, an algorithm that explores *closed* subvectors to find significant ones. We prove that ClosedVect is optimal in terms of the number of search states.

We validate the quality of our technique through experiments on chemical compounds and synthetic graphs. In particular, we find that a specific subgraph, neither largest nor

most frequent, turns out to be the largest common subgraph in a specific class of medically effective compounds. This finding validates the practical usefulness of our approach. We also demonstrate the efficiency of the computational methods and the feature vector mining algorithm.

The main contributions of our work are as follows:

1. We propose a technique for computing the p-values of frequent subgraphs, and show that frequent subgraph can be ranked by this measure. Efficient methods are developed for computing p-values and lower bounds.

2. We address the problem of feature vector mining, and present an algorithm for mining significant closed subvectors. This is an important problem in its own right.

The remainder of the paper is organized as follows. Section 2 discusses how to represent graphs as feature vectors. Section 3 presents the probabilistic model. Section 4 describes methods for computing p-values and lower bounds. Section 5 describes a simplified probabilistic model. Section 6 describes feature vector mining. Experimental results are reported in Section 7. Section 8 discusses related work. We conclude with a brief discussion in Section 9.

2 Representing Graphs as Feature Vectors

We view a graph as a collection of basis elements $\mathbb{B} = \{\hat{b}_1, ..., \hat{b}_m\}$. These basis elements can be vertices, edges, or small graphs. Each basis element $\hat{b}_i$ is associated with a *prior probability* θ_i. We first discuss how to select basis elements and transform graphs into feature vectors.

2.1 Feature Selection

The selection of basis elements is application-dependent and may require domain knowledge. A basic approach is to select all types of vertices or edges as features. This approach can be done efficiently and the meaning is clear: we evaluate the statistical significance of a subgraph by looking at how its vertices or edges are distributed. The drawback of this approach is that it does not capture any structural information of graphs.

For other graphs such as chemical compounds, one may choose small graphs such as Benzene rings. In this case, the number of available elements may grow dramatically, and these small graphs may overlap structurally. Thus, selecting a representative subset would be more appropriate. The following criteria for selection can be used: 1) frequency: frequent basis elements are more representative of graphs; 2) size: large basis elements carry more structural information (but would be less frequent); 3) structural overlap: overlapping basis elements are relatively not independent; 4) Co-occurrence: basis elements that frequently occur together are relatively not independent.

Generally, it is computationally difficult to select the optimal subset of basis elements. One may simply use a greedy approach [16]: choose the k^{th} best element according to its benefit gained (e.g., frequency) and its relevance (e.g., overlap, covariance) to the previously selected $k-1$ basis elements:

$$\begin{aligned} b_1 &= \arg\max_b \{w_1 freq(b) + w_2 size(b)\} \\ b_k &= \arg\max_b \{w_1 freq(b) + w_2 size(b) \\ &- \frac{w_3}{k-1}\sum_{i=1}^{k-1} sim(b_i, b) - \frac{w_4}{k-1}\sum_{i=1}^{k-1} cov(b_i, b)\}, \\ & k = 2, ..., m \end{aligned} \quad (1)$$

where $w_1 - w_4$ are weighting factors, $freq(b)$ is the frequency of b, $size(b)$ is the size of b, $sim(b_i, b)$ is the overlap between b_i and b, and $cov(b_i, b)$ is the covariance between b_i and b. All terms are normalized to $[0, 1]$. The procedure repeats until m features are selected.

For the sample database shown in Figure 1, we use all kinds of edges as the basis, i.e., $\mathbb{B}$={A-B, A-C, B-B, B-C, C-C}. The prior probabilities are empirically computed using their frequency in the database, i.e., $\underline{\theta} = (\frac{6}{17}, \frac{2}{17}, \frac{3}{17}, \frac{5}{17}, \frac{1}{17})$.

2.2 Transforming Graphs into Feature Vectors

After a basis is selected, we transform (sub)graphs into feature vectors. We denote a feature vector by $\underline{x} = (x_1, ..., x_m)$, where x_i counts the frequency of feature $\hat{b}_i$ in the graph. The size of $\underline{x}$ is defined as $|\underline{x}| = \sum x_i$. Vector $\underline{x}$ is a *sub-vector* of vector $\underline{y}$ (and $\underline{y}$ a *super-vector* of $\underline{x}$) if $x_i \leq y_i$ for $i = 1, ..., m$, and is denoted by $\underline{x} \subseteq \underline{y}$. The *floor* of two vectors $\underline{x}$ and $\underline{y}$ is a vector $\underline{v}$ where $v_i = min(x_i, y_i)$ for $i = 1, ..., m$. The definition extends to a group of vectors. The *ceiling* of a group of vectors is defined analogously.

For the sample subgraphs shown in Figure 2, Table 1 shows their corresponding feature vectors.

	A-B	A-C	B-B	B-C	C-C
g_1	1	0	0	0	0
g_2	1	0	1	0	0
g_3	2	0	1	0	0
g_4	1	0	1	1	0
g_5	2	0	1	1	0

Table 1. Feature vectors of the subgraphs in Figure 2

3 Probabilistic Model

In this section, we model the probability with which a feature vector $\underline{x}$ (corresponding to a subgraph) occurs in a random vector (corresponding to a random graph) obtained using prior probabilities on the basis elements, and the probability distribution of $\underline{x}$'s support in a database of random vectors. Statistical significance is obtained by comparison to its observed support.

3.1 Probability of occurrence of a feature vector in a random vector

We regard the basis $\mathbb{B}$ as a set of m distinct events, one for every basis element, where basis element $\hat{b}_i$ is associated with its prior probability θ_i. A feature vector of a certain size ℓ is thus regarded as an outcome of ℓ independent trials.

Given a feature vector $\underline{y} = (y_1, ..., y_m)$, $|\underline{y}| = \ell$, the probability that $\underline{y}$ is observed in ℓ trials can be modeled by a multinomial distribution:

$$Q(\underline{y}) = \frac{\ell!}{\prod y_i!} \prod_{i=1}^{m} \theta_i^{y_i}, \tag{2}$$

In other words, Eqn. (2) gives the probability of observing $\underline{y}$ in a random vector of size ℓ.

Let $\underline{x}$ be the feature vector of a subgraph g. Then, the probability that $\underline{x}$ occurs in a random vector of size ℓ is a cumulative mass function (c.m.f.) of Eqn. (2):

$$P(\underline{x}; \ell) = \sum_{\underline{y} s.t. y_i \geq x_i, |\underline{y}|=\ell} Q(\underline{y}) \tag{3}$$

In other words, this is the probability that $\underline{x}$ occurs in a random vector of size ℓ. The size constraint ℓ is reasonable: the larger a random vector, the more likely that $\underline{x}$ will occur in the vector.

For example, the feature vector of subgraph g_3 in Figure 2 is $\underline{x} = (2, 0, 1, 0, 0)$. The probability that $\underline{x}$ occurs in a random vector of size 3 is $P(\underline{x}; 3) = 0.066$.

The computation of Eqn. (3) is not trivial when m and ℓ are large. We will discuss an efficient way to compute Eqn. (3) in Section 4.

3.2 Probability distribution of a feature vector's support in a database of random vectors

Now we consider the support of $\underline{x}$ in the context of a database of random vectors. This support is a random variable that follows a probability distribution. Since we are assessing the significance of $\underline{x}$ in a given target database, the random database should have the same number of vectors as the target database, and vectors in the random database should have similar sizes as those in the target database. Let n be the number of vectors in the target database, we summarize the sizes of the vectors by $\underline{\ell} = (\ell_1, ..., \ell_d)$ and $\underline{n} = (n_1, ..., n_d)$, where n_i is the number of vectors of size ℓ_i, and $\sum n_i = n$.

If we regard a random vector as a trial, and the occurrence of $\underline{x}$ in the vector as a "success". Then, the database of random vectors corresponds to n trials, and the support of $\underline{x}$ corresponds to the number of successes in n trials. If the sizes of the vectors were identical, say ℓ, then the support can be modeled as a binomial random variable, with parameters n and $P(\underline{x}; \ell)$. When the sizes are distinct, each size will correspond to one binomial random variable with parameters n_i and $P(\underline{x}; \ell_i)$. Then, the support of $\underline{x}$ is the sum of the binomial random variables: the probability of $\underline{x}$'s support being equal to μ is given by

$$R(\mu; \underline{x}, \underline{\ell}, \underline{n}) = \sum_{\sum t_j = \mu}^{d} bino(t_j; n_i, P(\underline{x}; \ell_i)) \tag{4}$$

where $bino(t; n, p) = \binom{n}{t} p^t (1-p)^{n-t}$ is the binomial probability distribution. In other words, the j^{th} binomial contributes t_j successes, with the sum of them equal to μ. All possible combinations of t_j give the total probability of observing μ.

For the sample database of Figure 1, a random database would have $\underline{\ell} = (3, 4)$ and $\underline{n} = (3, 2)$. Figure 5 plots the probability distribution of subgraph g_3's support in the random database.

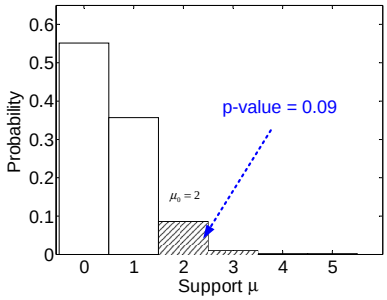


Figure 5. Probability distribution of g_3's support and its p-value

We will discuss an efficient method of computing Eqn. (4) in Section 4.

3.3 Statistical Significance of a Feature Vector

Let μ_0 be the observed support in the target database. Then, the p-value, i.e., the probability of observing a support of at least μ_0 in the random database is given by

$$R(\mu \geq \mu_0; \underline{x}, \underline{\ell}, \underline{n}) = \sum_{\mu=\mu_0}^{n} R(\mu; \underline{x}, \underline{\ell}, \underline{n}). \tag{5}$$

We denote Eqn. (5) by *p-value*($\underline{x}$, μ_0). The smaller the p-value, the more statistically significant is the feature vector.

For example, g_3's observed support is 2. Its p-value is shown as the shaded part in Figure 5.

The p-value has the following monotonicity properties:

1. Given two vectors $\underline{x_1}$ and $\underline{x_2}$, if $\underline{x_1} \subseteq \underline{x_2}$, then *p-value*($\underline{x_1}$, μ_0)$\geq$ *p-value*($\underline{x_2}$, μ_0) for any μ_0.

2. Given two supports μ_1 and μ_2, if $\mu_1 \leq \mu_2$, then *p-value*($\underline{x}$, μ_1)$\geq$ *p-value*($\underline{x}$, μ_2) for any $\underline{x}$.

A frequent pattern is *closed* if none of its super-patterns has the same support as the pattern. According to the monotonicity properties, the p-value of a non-closed pattern is greater than or equal to that of its closed super-pattern. Thus, we can consider only closed sub-vectors/subgraphs.

Now, we are ready to answer the question regarding significance raised in Figure 2. The p-value of each subgraph is computed and shown in Table 2. Their expected supports are computed as well. Among the subgraphs listed in Figure 2, g_3 has the smallest p-value. Thus, we can claim that g_3 is the most statistically significant (though it is neither the largest nor the most frequent).

	$\bar{\mu}$	μ_0	p-value
g_1	3.84	4	0.67
g_2	1.65	3	0.20
g_3	**0.55**	**2**	**0.09**
g_4	0.85	2	0.20
g_5	0.16	1	0.15

Table 2. P-values of the subgraphs in Figure 2; subgraph g_3 has the smallest p-value.

4 Computation of P-values and Lower Bounds

In this section, we present efficient methods to compute Eqn. (3) and Eqn. (4), which would take exponential time using naive approaches. Lower bounds are also developed for fast estimation.

4.1 Computation of $P(\underline{x}; \ell)$

We develop a divide-and-conquer scheme for the computation of $P(\underline{x}; \ell)$, the probability that $\underline{x}$ occurs in a vector of size ℓ (Eqn. (3)). The idea is to split $\underline{x}$ into two halves: $\underline{x_1} = (x_1, ..., x_{\frac{m}{2}})$ and $\underline{x_2} = (x_{\frac{m}{2}+1}, ..., x_m)$, and then take the convolution of $\underline{x_1}$ and $\underline{x_2}$. In other words, the probability of observing $\underline{x}$ in a vector of size ℓ is equal to the probability of observing $\underline{x_1}$ in the first half, and observing $\underline{x_2}$ in the second half of the vector, provided that the sum of the two halves is equal to ℓ. The recurrence relation is given by

$$P(\underline{x}; \ell) = \sum_{t=|\underline{x_1}|}^{\ell-|\underline{x_2}|} \binom{\ell}{t} P(\underline{x_1}; t) P(\underline{x_2}; \ell - t) \qquad (6)$$

The splitting is done recursively until a single bin x_i is reached, where $P(x_i; \ell) = \theta_i^\ell$. The recurrence form requires an array for $P(\underline{x_1}; t)$ and $P(\underline{x_2}; \ell - t)$ respectively. Thus, at each recurrence step we need to compute an array $P(\underline{x}; s)$ for $s = |\underline{x}|, ..., \ell$.

The time complexity of the above recurrence is analyzed as follows. Let $f(m, \ell)$ be the time to compute array $P(\underline{x}; s)$ for $s = |\underline{x}|, ..., \ell$, then it takes $2f(\frac{m}{2}, \ell)$ to compute the arrays of the sub-units, and $(\ell - |\underline{x}|)^2$ to compute the array of the current unit. Thus, $f(m, \ell) = O(2f(\frac{m}{2}, \ell) + (\ell - |\underline{x}|)^2)$. Solving the equation yields the time complexity of $O(m(\ell - |\underline{x}|)^2)$. Value m can be further reduced to the number of non-zero bins in $\underline{x}$.

4.2 Computation of Sum of Binomials

The sum of binomial distributions (Eqn. (4)) can also be computed using a divide-and-conquer scheme. To observe μ in the d binomials, one observes t in the first $\frac{d}{2}$ binomials and $\mu - t$ in the rest binomials for all $t = 0, ..., \mu$. The recurrence is given by

$$R(\mu; \underline{x}, \underline{\ell}, \underline{n}) = \sum_{t=0}^{\mu} R(t; \underline{x}, \underline{\ell_1}, \underline{n_1}) R(\mu; \underline{x}, \underline{\ell_2}, \underline{n_2}) \qquad (7)$$

where $\langle \underline{\ell_1}, \underline{n_1} \rangle$ and $\langle \underline{\ell_2}, \underline{n_2} \rangle$ correspond to the first and the second half of the d binomials, respectively. Analogous to Subsection 4.1, the time complexity can be computed to be $O(d\mu^2)$.

4.3 Lower Bound to P-value

For faster estimation, we develop a lower bound to the p-value. The lower bound to the p-value is obtained through a lower bound to $P(\underline{x}; \ell)$. Theorem 1 shows how this is done.

Theorem 1. *Let $\underline{x}$ and $\underline{y}$ be two vectors, if $P(\underline{x}; \ell) \leq P(\underline{y}, \ell)$ for $\forall \ell$, then* p-value($\underline{x}$, μ) $\leq$ p-value($\underline{y}$, μ) *for $\forall \mu$.*

Next, we get a lower bound to $P(\underline{x}; \ell)$ by decoupling the multinomial into a product of binomials.

Theorem 2. *(Lower bound to $P(\underline{x}; \ell)$)*

$$\begin{aligned} P(\underline{x}; \ell) &\geq \prod_{i=1}^{m} \sum_{t=x_i}^{a_i} \binom{a_i}{t} \theta_i^t (1 - \theta_i)^{a_i - t} \\ &= \prod_{i=1}^{m} I(\theta_i; x_i, a_i - x_i + 1) \end{aligned} \qquad (8)$$

where $a_i = \ell - \sum_{t=1}^{i-1} x_i$*, and* $I(\theta_i; x_i, a_i - x_i + 1)$ *is the regularized Beta function.*

Proof. Let us start with the simple case where X has two bins x_1 and x_2. We rewrite $P(X, N)$ in the form of $P(y_1 \geq x_1, y_2 \geq x_2; N)$, i.e., the probability of observing at least x_1 in the first bin and at least x_2 in the second bin in N trials. Note that $P(y_1 \geq x_1, y_2 \geq x_2; N) = P(y_1 \geq x_1; N)P(y_2 \geq x_2; N|y_1 \geq x_1; N)$. For $P(y_2 \geq x_2; N|y_1 \geq x_1; N)$, we can always ensure the condition $y_1 \geq x_1$ by reserving x_1 trials from the N trials. Thus, $P(y_2 \geq x_2; N|y_1 \geq x_1; N) \geq P(y_2 \geq x_2; N - x_1)$. Hence $P(y_1 \geq x_1, y_2 \geq x_2; N) \geq P(y_1 \geq x_1; N)P(y_2 \geq x_2; N - x_1)$. When we extend the above reasoning to more than two bins, we obtain the product of sums in Eqn. (8). Equivalence to the regularized Beta function is known from the statistics literature[1]. □

Theorem 3 gives an upper bound to $P(\underline{x}; \ell)$ in an analogous manner.

Theorem 3. *(Upper bound to* $P(\underline{x}; \ell)$*)*

$$\begin{aligned} P(\underline{x}; \ell) &\leq \prod_{i=1}^{m} \sum_{t=x_i}^{\ell} \binom{\ell}{t} \theta_i^t (1-\theta_i)^{\ell-t} \\ &= \prod_{i=1}^{m} I(\theta_i; x_i, \ell - x_i + 1) \end{aligned} \tag{9}$$

5 A Simplified Model

The computation of p-values and lower bounds as illustrated in the previous section does not scale to very large databases. In this section, we present a simplified model in which the computation of p-values is much more efficient.

In our previous model, we had a constraint on the size of random vectors. Our first simplification is to relax this constraint, and consider the probability that a feature vector occurs in a random vector of arbitrary size. The probability can be written as

$$P(\underline{x}) = P(Y_1 \geq x_1, ..., Y_m \geq x_m) \tag{10}$$

Further, if we assume that different types of basis elements are orthogonal, then the above joint probability can be decoupled into a product of probabilities:

$$\widehat{P}(\underline{x}) = \prod_{i=1}^{m} P(Y_i \geq x_i) \tag{11}$$

where $P(Y_i \geq x_i)$ is the probability that element $\hat{b}_i$ occurs at least x_i times in a random vector.

[1] http://mathworld.wolfram.com/BinomialDistribution.html

Since $\widehat{P}(\underline{x})$ is fixed, the support of $\underline{x}$ in a database of random vectors can be modeled by a single binomial distribution, with parameters n and $\widehat{P}(\underline{x})$.

Under the simplified model, we compute the p-value as follows.

1. Empirically obtain the prior probabilities $P(Y_i \geq j)$ for every basis element$\hat{b}_i$ and every j (up to the maximum possible value).

   For example, element $\hat{b}_1$ ="A-B" occurs twice (G_1 and G_2) in the sample database, thus $P(Y_1 \geq 2) = \frac{2}{5}$. Similarly, $P(Y_1 \geq 1) = \frac{4}{5}$, $P(Y_1 \geq 0) = 1$, $P(Y_3 \geq 1) = \frac{3}{5}$, etc.

2. Compute $\widehat{P}(\underline{x})$ using Eqn. (11). For subgraph g_3, $\underline{x} = (2, 0, 1, 0, 0)$. Thus $\widehat{P}(\underline{x}) = P(Y_1 \geq 2) \times P(Y_3 \geq 1) = \frac{2}{5} \times \frac{3}{5} = \frac{6}{25}$.

3. Compute the p-value of $\underline{x}$ by $\sum_{\mu_0}^{n} bino(\mu; n, \widehat{P}(\underline{x}))$, or equivalently by the regularized Beta function $I(\widehat{P}(\underline{x}); \mu_0, n)$. When both $n\widehat{P}(\underline{x})$ and $n(1 - \widehat{P}(\underline{x}))$ are large, the binomial distribution can be approximated by a normal distribution.

6 Feature Vector Mining

As frequent subgraphs are represented as feature vectors and evaluated for statistical significance, an interesting question arises: *can we directly search top-K significant sub-vectors, or sub-vectors above a significance threshold?* To our best knowledge, the problem of feature vector mining has not been addressed before. Feature vector mining is important in two aspects. First, feature vectors, also known as histograms and multisets, are common ways to summarize complex data. As a result, feature vector patterns are profiles of structured patterns, and feature vector mining can work as a foundation of structured pattern mining. Second, feature vector mining is an important generalization of the well studied frequent itemset mining: each item is now associated with a count instead of a boolean value.

We develop *ClosedVect*, an algorithm that explores frequent *closed* sub-vectors to find significant ones. The algorithm consists of two phases: exploring closed sub-vectors and evaluating the significance of a closed sub-vector.

6.1 Exploring Closed Sub-Vectors

Alg. 1 outlines the phase of exploring closed sub-vectors. The algorithm explores sub-vectors in a bottom-up, depth-first manner. At each search state, the algorithm "jumps" to a future state that has an immediately smaller supporting set along a branch (line 3). The corresponding sub-vector is then promoted as the *floor* of the supporting set (line 6).

To prevent duplicates, each state is associated with a beginning position b. Any future state must extend at a position greater than or equal to b. All extensions starting at the same position are placed along the same search branch. If an extension designated at position i results in a starting position of less than i, then it must be a duplicate extension (lines 7-8).

The evaluation phase (line 1) computes the p-value of a sub-vector and reports top-K significant ones. Lines 9-10 estimate a lower bound on the p-value of the super-vectors of $\underline{x}'$ and prune it if this bound is too high. The evaluation phase and the pruning will be discussed in Subsection 6.2 and 6.3.

Alg. 1 ClosedVect($\underline{x}$, $\mathbb{S}$, b)

$\underline{x}$: current sub-vector;
$\mathbb{S}$: supporting set of $\underline{x}$, i.e., feature vectors in the database that contain $\underline{x}$;
b: beginning position at which bins can be extended;

```
1:  Eval(x, |S|);
2:  for i := b to m do
3:      S' ← {Y | Y ∈ S, Y_i > x_i};
4:      if |S'| <minSupport then
5:          continue;
6:      x' := floor(S');
7:      if ∃j < i such that x'_j > x_j then
8:          continue;
9:      if p-value(ceiling(S'), |S'|) ≥ maxPvalue then
10:         continue;
11:     ClosedVect(x', S', i);
```

Figure 6 shows a running example of Alg 1. The underlined numbers denote the beginning position b of each state. Duplicate search states are pruned by examining the search order. For example, an extension to state "2 $\underline{3}$ 2" at position "$\underline{3}$" leads to a supporting set "$\{h_1, h_3\}$", of which the *floor* is "3 $\underline{4}$ 2". However, this extension violates the search order and is pruned (lines 7-8).

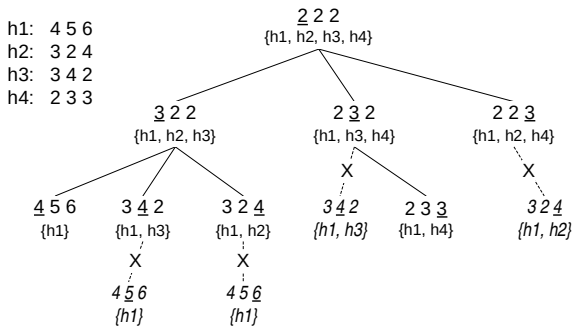


Figure 6. A running example of ClosedVect

Now, we show the correctness and efficiency of algorithm ClosedVect. We say that an algorithm is *complete* if it explores all desired answers. It is *compact* if every search state finds at least one distinct answer. It is *duplicate-free* if it does not extend duplicate search states nor generate duplicate answers.

First, we state a lemma that establishes the use of *floor* of supporting sets when exploring closed sub-vectors.

Lemma 1. *For any closed sub-vector $\underline{x}$ and its supporting set $\mathbb{S}$, $\underline{x} = floor(\mathbb{S})$.*

Theorem 4. *(Correctness and Efficiency of ClosedVect) Algorithm ClosedVect explores closed and only closed sub-vectors. It is complete, compact, and duplicate-free.*

Proof. 1) *completeness*: For any closed sub-vector $\underline{x}$, let $\mathbb{S}$ be its supporting set, then $\underline{x} = floor(\mathbb{S})$. We show that $\underline{x}$ can be found in a search state. Starting from the root state $\underline{r}$, let i be the first bin such that $floor(\mathbb{S})_i > r_i$, then $\mathbb{S}$ must be a subset of the supporting set at extension i. By induction on the number of bins, the supporting set will eventually shrink to $\mathbb{S}$, hence $\underline{x}$ is found. 2) *compactness*: By construction, each state in the search tree corresponds to a closed sub-vector. 3) *duplicate-free*: According to the search order, each search state can be uniquely located in the search tree. □

In other words, ClosedVect is optimal in terms of the number of search states because every search state corresponds to a distinct closed sub-vector.

6.2 Evaluating Closed Frequent Sub-Vectors

Next, we describe the evaluation phase of our feature mining algorithm. Alg. 2 outlines the evaluation phase. A priority queue is used to maintain the answer set, i.e., top-K significant sub-vectors found so far. The p-value threshold *maxPvalue* is the p-value of the K^{th} significant sub-vector found so far. To evaluate a candidate sub-vector, the lower bound to the p-value (Eqn. (8)) is examined before the computation of exact p-values (Eqn. (5)). If they are both less than *maxPvalue*, then both the priority queue and *maxPvalue* are updated.

To search top-K significant sub-vectors, one sets the initial maxPvalue as 1; to search sub-vectors above a significance threshold, one sets $K = +\infty$.

6.3 Lower Bound of P-values of Super-Vectors

Next, we study how to compute a lower bound to the p-values of all super-vectors of a given sub-vector. This is used to prune unnecessary extensions in algorithm ClosedVect (lines 9-10). There are two approaches to computing this lower bound. The first approach computes the *ceiling* of the supporting set and uses it to bound the p-value.

Alg. 2 Eval($\underline{x}$, μ_0)

$\underline{x}$: a sub-vector;
μ_0: support of $\underline{x}$;
PQ: Priority queue for top-K answers;

if *p-value_lowerbound*($\underline{x}$, μ_0)<*maxPvalue* **then**
  if *p-value*($\underline{x}$, μ_0)< *maxPvalue* **then**
    Insert $\langle$*p-value*($\underline{x}$, μ_0), $\underline{x}\rangle$ into PQ;
    if $|PQ| > K$ **then**
      Pop an item from PQ;
      maxPvalue:=PQ.top.pvalue;

Theorem 5. *Let $\underline{x}$ and $\underline{u}$ be two vectors and $\underline{x} \subset \underline{u}$, then for any $\underline{y}$ subject to $\underline{x} \subseteq \underline{y} \subseteq \underline{u}$, p-value($\underline{y}$, support($\underline{y}$)) $\geq$ p-value($\underline{u}$, support($\underline{x}$)).*

Proof. The proof follows from the monotonicity property of the p-value. □

The second approach constructs the most skewed super-vector of a certain size from $\underline{x}$, and uses it to bound the p-value of all super-vectors of the same size. The following lemma allows us to incrementally skew a vector.

Lemma 2. *Assuming $\theta_1 \leq \theta_2 \leq ... \leq \theta_m$. Let $\underline{x}$=(x_1, ..., x_i, ..., x_j, ..., x_m).*

(1) If $x_i \geq x_j$, let $\underline{x}' = (x_1, ..., x_i+1, ..., x_j-1, ..., x_m)$, then $P(\underline{x}'; \ell) \leq P(\underline{x}; \ell)$. In this case, we increment x_i and decrement x_j.

(2) If $x_i < x_j$, let $\underline{x}' = (x_1, ..., x_j, ..., x_i, ..., x_m)$., then $P(\underline{x}'; \ell) \leq P(\underline{x}; \ell)$. In this case, we switch x_i and x_j.

Proof. See Appendix A. □

In the following theorem, we estimate a lower bound to the p-value of a super-vector of constant size $|\underline{x}| + \delta$ by first skewing $\underline{x}$ to $\underline{x}'$ using Lemma 2, and then adding δ to bins with the smallest prior probabilities.

Theorem 6. *Assuming $\theta_1 \leq \theta_2 \leq ... \leq \theta_m$. Let $\underline{u} = (u_1, ..., u_m)$ be the ceiling of super-vectors, $u_1 \geq u_2 \geq ... \geq u_m$. Given $\underline{x}$, $\underline{x} \subseteq \underline{u}$, sort $x'_i s$ in non-increasing order: $\underline{x}' = (x'_1, ..., x'_m)$, $x'_1 \geq x'_2 \geq ... \geq x'_m$. Given $\delta > 0$, construct $\underline{y_m}$ from $\underline{x}'$ as follows: fill x'_1 up to u_1, then x'_2 up to u_2, ..., and so on until δ is used up. Then, for any $\underline{y}$ subject to $\underline{x} \subseteq \underline{y} \subseteq \underline{u}$ and $|\underline{y}| = |\underline{x}|+\delta$, $P(\underline{y}; \ell) \geq P(\underline{y_m}; \ell)$.*

Proof. According to Lemma 2, $\underline{y}$ can be iteratively transformed into $\underline{y_m}$ with $P(\underline{y}; \ell)$ non-increasing. □

Theorems 5 and 6 are intended to prune search states where the closed sub-vectors are large. They can be especially effective if large sub-vectors are not significant.

7 Experimental Results

In this section, we report experimental results that validate the quality and efficiency of the proposed techniques. The experiments are divided into two parts: 1) Validation of the quality of our probabilistic model, and 2) Performance evaluation of the feature vector mining algorithm as well as the p-value computation.

Three datasets are used in our experiments. The first dataset is the DTP-AIDS Antiviral Screen chemical compound dataset from NCI/NIH[2]. The compounds have been classified into three categories according to their AIDS antiviral activities. We focus on the category of confirmed active (CA) which contains 422 chemical compounds. On average, each graph has 40 vertices and 42 edges. The second dataset is synthetic graphs for recall tests. The third dataset is a web page visits dataset.

The p-value computation and the feature vector mining algorithm were implemented in Java using Sun JDK 1.5.0. The regularized Beta function (Eqn. (8)) was computed using Apache's Commons-Math Library[3]. All experiments were performed on an Intel 2.8GHz, 1G memory running MS Windows XP Professional.

We use CloseGraph [10] to find frequent closed subgraphs. We compare the p-value ranking with a simple ranking approach based on size. To our best knowledge, there are no other methods that evaluate the statistical significance of frequent subgraphs in a graph database. Thus, comparative assessments to other statistical methods are not presented.

7.1 Evaluating the Quality of the Results

7.1.1 Chemical Compound Graphs

We demonstrate the practical usefulness of our method on the chemical compound dataset. Two sets of basis elements are generated to transform subgraphs into feature vectors. The first set of basis elements consists of all different edges (39 in total), namely *1-edge basis*. For the second set of basis elements, we consider all possible subgraphs containing 3 edges (322 in total), and select 30 of them using the greedy approach discussed in Section 2. We call this the *3-edge basis*. For each case, we compute the prior probabilities using their frequency in the background dataset (containing around 42,000 compounds).

Using CloseGraph [10] with minimum support *minSup*=5%, 7879 closed subgraphs are generated [4]. For each of them, we compute its p-value using the two bases and the two models for p-value computation (exact and simplified).

[2] http://dtp.nci.nih.gov/

[3] http://jakarta.apache.org/commons/math/

[4] the results are different from [10] since aromatic bonds are not specially treated.

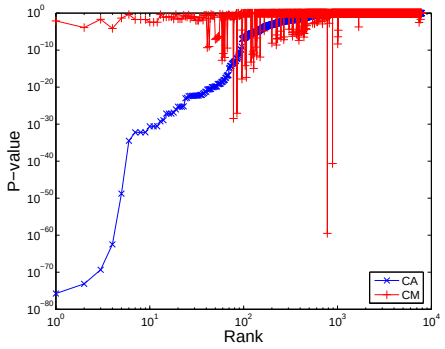


Figure 7. P-value vs. rank with cross-validation

Figure 7 shows the p-values of the subgraphs vs. their ranks using 3-edge basis and the exact model. To cross-validate the significance of the subgraphs, we also compute their p-values in the category of confirmed moderately active (CM) for comparison. As shown in the figure, the p-values of the discovered subgraphs are much smaller than they would be in the context of the CM category. Further, a large number of the subgraphs are statistically insignificant. Using a p-value cutoff, say 0.01, we are able to reduce the number of discovered subgraphs by one order of magnitude.

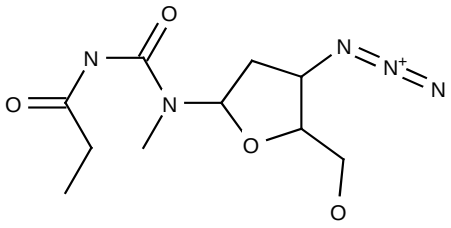
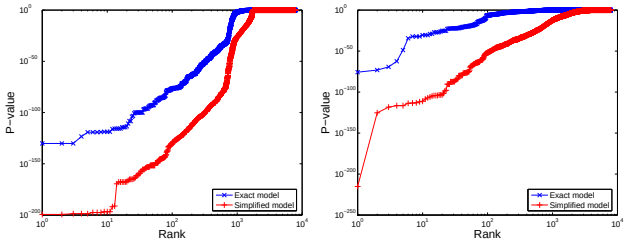


Figure 8. The most significant subgraph in CA

Figure 8 shows the most significant subgraph found in our results (the unlabeled nodes are C atoms). It is ranked 1^{st} in both the exact and simplified model using the 3-edge basis. This subgraph has 19 edges and 15% support. We found that this subgraph is the largest common subgraph in the chemical class of Azido Pyrimidines [5]. The AZT compound (NSC 602670), a super graph of this subgraph, has an extra edge on the left hexagon and 12% support. It is ranked 3^{rd} in the exact model and 2^{nd} in the simplified model. The compound has been widely used for HIV inhibition. The findings validate the practical usefulness of our approach.

Figure 9 shows the p-values of the subgraphs vs. their ranks using different feature bases and different models for the computation of p-values. The p-values in the simplified

[5]http://dtp.nci.nih.gov/docs/aids/searches/list.html



(a) 1-edge basis (b) 3-edge basis

Figure 9. P-value vs. rank with different feature bases and models

model are smaller than those in the exact model. The underlying reason is the stronger assumption by the simplified model that different basis elements are totally independent, whereas in the exact model, they are constrained by the size of random graphs. Nevertheless, the rankings by the two models are more or less consistent. Under the 3-edge basis, for example, the top 10 subgraphs of the exact model show up in the top 30 subgraphs of the simplified model.

We compare our ranking approach with a naive ranking approach: rank by size (in case of tie, rank by support). Table 3 shows the rankings of some special subgraphs: the most significant subgraph (AZT*), the largest subgraph, and Benzene (a ring with six carbons). There is no current scientific evidence regarding the importance of the largest subgraph. As shown in the table, ranking by p-value is much more appropriate than the ranking by size. And the most significant subgraph is not necessarily the largest or the most frequent subgraph.

Subgraph	Support	Size	Rank by p-value				Rank by size
			3-edge basis		1-edge basis		
			strict	simpl.	strict	simpl.	
AZT*	15%	19	1^{st}	1^{st}	40^{th}	69^{th}	428^{th}
Largest	5%	34	914^{th}	142^{nd}	752^{nd}	751^{st}	1^{st}
Benzene	70%	6	886^{th}	1424^{th}	6820^{th}	1875^{th}	6969^{th}

Table 3. Ranking by different approaches

7.1.2 Recall Tests on Synthetic graphs

We also verify the quality of our method through recall tests on synthetic graphs. The procedure of recall tests is illustrated in Figure 10. The basic idea is to embed some significant subgraphs into a synthetic database, and then see how they can be recalled through p-value ranking. The tests are "supervised" in that all prior knowledge, such as basis elements and significant subgraphs, are known in advance.

We generate synthetic graphs as follows. Let L_V and L_E be the label set of vertices and edges respectively. The size of a graph is measured by the number of edges. First, we

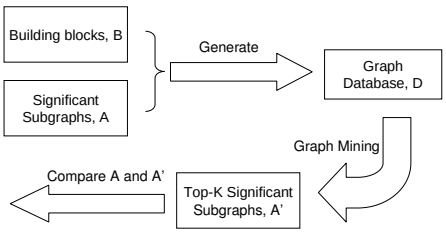


Figure 10. Recall tests on synthetic graphs

generate a set of building blocks (also the basis elements) $\mathbb{B} = \{\hat{b}_1, ..., \hat{b}_m\}$. Each building block is a tiny subgraph generated by randomly adding an edge to the subgraph until it reaches a given fixed size zB. Then, we generate k significant subgraphs $\mathbb{A} = \{A_1, ..., A_k\}$ using the building blocks. Each significant subgraph is generated by randomly inserting a building block into the subgraph until it reaches size zA, which has a Poisson distribution. Next, we use $\mathbb{B}$ and $\mathbb{A}$ to generate the database graphs. Each database graph has a probability of $P_{\mathbb{A}}$ of selecting a significant subgraph from $\mathbb{A}$. Then, the building blocks are randomly selected and inserted into the database graph until it reaches size zG, which has a Poisson distribution.

Next, we use our technique to discover the frequent subgraphs, compute their p-values, and see how they are ranked.

In our experiments, we fix $|L_V| = |L_E| = 10$, $zB = 3$, $m = 100$, $zA = 10$, $k = 5$, $|\mathbb{B}| = 100$, $zG = 30$, $|\mathbb{D}| = 1000$, and $P_{\mathbb{A}} = 0.6 \sim 1.0$.

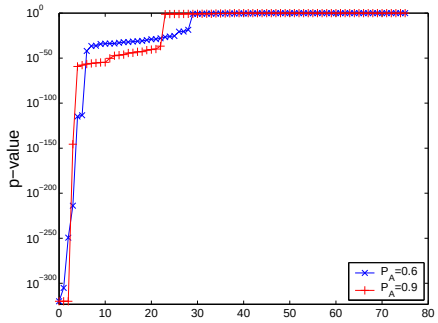


Figure 11. P-value vs. rank

Figure 11 shows the p-value vs. rank for $P_{\mathbb{A}}$=0.6 and 0.9 respectively. On average, the p-values for $P_{\mathbb{A}} = 0.9$ are smaller than those for $P_{\mathbb{A}} = 0.6$. This is because for $P_{\mathbb{A}} = 0.9$, database graphs contains more significant subgraphs.

Table 4 shows the rankings of the subgraphs in $\mathbb{A}$. All significant subgraphs in $\mathbb{A}$ have been discovered and ranked at very high positions.

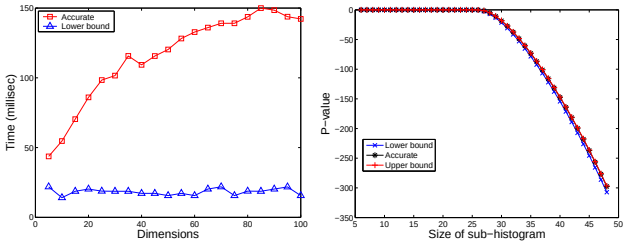
	Rankings of significant subgraphs
P_A=0.6	0, 1, 2, 11, 23
P_A=0.9	0, 1, 2, 11, 13

Table 4. Rankings of subgraphs in $\mathbb{A}$

7.2 Computation Costs of P-values and Lower Bounds

We evaluate the computation costs and lower bounds of p-values, as well as tightness of lower bounds (Section 4) using random data. The scenario is set up as follows. The size of the database is 1000; the sizes of database vectors randomly range from 50 to 300; the number of distinct sizes of database vectors is 100, i.e., there are 100 binomial distributions; the number of dimensions of vectors ranges from 5 to 100 with an interval of 5; for each number of dimensions, we randomly generate 50 sub-vectors of size 30, compute their p-values and lower bounds, and average the running times.

Figure 12(a) shows the running time for a single p-value computation in the number of dimensions. 'Accurate' refers to the computation of exact p-values; 'Lower bound' refers to that of lower bounds (Eqn. (8)). The running time for exact computation increases when the number of dimensions increases. The time complexity is actually linear in m', the number of non-zero bins in the sub-vectors. The running time for the lower bound computations is much less than that of the accurate computation.



(a) Running time (b) Lower/upper bounds

Figure 12. Computation costs and bounds of p-values

To evaluate the tightness of the lower bound and upper bound, we gradually grow a sub-vector starting with size 5 until size 50, and compute the p-value and lower/upper bounds. This procedure fits the typical depth-first feature vector mining scenario.

Figure 12(b) shows the lower bound and upper bound to the p-value. Both the lower bound and the upper bound are close to the exact p-value in orders of magnitude. Thus, they can be effectively used to estimate the exact p-value.

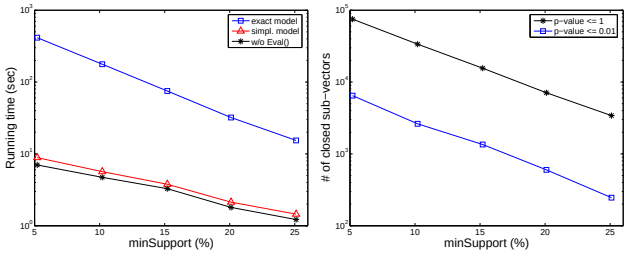
7.3 Feature Vector Mining

7.3.1 Chemical Compound Graphs

We evaluate the performance of algorithm ClosedVect (Section 6) using the chemical compound dataset which contains 422 graphs. The graphs are transformed into feature vectors using the 3-edge basis. We run the algorithm using the exact model, the simplified model, and without p-value evaluation. The experimental settings are: minSupport=5$\sim$25%; K=+∞; maxPvalue=1, 0.01.

Figure 13(a) shows the running time of ClosedVect w.r.t. $minSupport$. As expected, the running time decreases with higher support thresholds. Also, the running time without p-value evaluation is only in seconds. This demonstrates the high efficiency of ClosedVect in exploring closed sub-vectors. With p-value computation, the simplified model adds a little amount of overhead. The exact model takes much longer in the computation of p-values. Actually, the computation time of a single p-value in the exact model is less than one second. It is the large number of closed sub-vectors that lead to the high running time.

Figure 13(b) shows the number of closed sub-vectors w.r.t. $minSupport$ under the exact model. With the maximum p-value threshold set at 0.01, the number of closed sub-vectors is reduced by one order of magnitude.



(a) Running time (b) # of closed sub-vectors

Figure 13. ClosedVect **on chemical compounds**

7.3.2 MSNBC Page Visits Data

We also run the ClosedVect algorithm on the MSNBC page visits dataset. The dataset is available in the UCI KDD archive repository[6]. It records the page visits of msnbc.com on a specific day. The dataset consists of 989,818 records, each of which is a sequence of page categories visited by a user. There are 17 page categories. Thus, each record in the data set is a vector of size 17 and we are interested in finding the sub-vectors that denote significant visit patterns.

[6]http://kdd.ics.uci.edu/databases/msnbc/msnbc.html

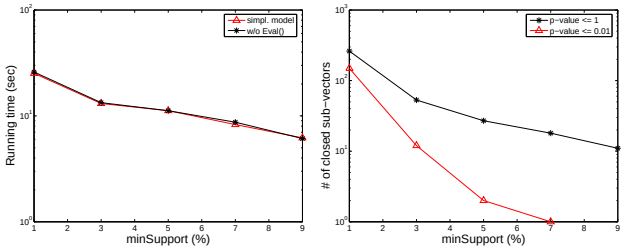
Table 5 shows the first few categories and the prior probabilities of being visited at least once (in the context of the simplified model).

Category	frontpage	news	tech	local	...
Prob. of ≥ 1 visit	0.316	0.177	0.123	0.123	...

Table 5. Page categories and prior probabilities

We use the simplified model to evaluate statistical significance. The experimental settings are: minSupport=1$\sim$9%; K=+∞; maxPvalue=1,0.01.

Figure 14(a) shows the running time w.r.t. $minSupport$. As shown in the figure, the ClosedVect algorithm is very efficient and scalable to large datasets (nearly 1 million records). Figure 14(b) shows the number of closed sub-vectors w.r.t. $minSupport$. The maximum p-value threshold effectively reduces the number of discovered results.



(a) Running time (b) # of Closed Sub-vectors

Figure 14. ClosedVect **on MSNBC page visits data**

A preliminary study of the results shows that the most significant sub-vectors are those with skewed distributions and at least two non-zero bins. For example, a discovered sub-vector with a high statistical significance was one in which users visited the *frontpage* seven times and *news* once; the corresponding support was only 1.1%. In contrast, another pattern in which users visited the *frontpage* eight times was not statistically significant, even though its support was 2.1%.

8 Related Work

Graph mining has been an active research topic recently. In the area of mining frequent subgraphs from a transactional graph database, Inokuchi et al. [7] addressed the problem using an Apriori approach. Kuramochi and Karypis [8] proposed FSG, an Apriori-based approach to frequent subgraph discovery. Yan and Han [9] proposed gSpan that efficiently explores frequent subgraphs. Their

later work [10] searched closed frequent subgraphs. Huan et al. [11] explored frequent subgraphs using a canonical adjacency matrix representation of graphs. Their later work [12] searched maximal frequent subgraphs. Vanetik et al. [13] proposed an Apriori approach using paths as building blocks. Their later work [14] addressed partially labeled graph patterns. In the area of mining frequent subtrees, Zaki [17] developed an algorithm to find frequent subtrees in a forest. Chi et al. [18] presented an index technique for free trees and applied it to frequent subtree mining. All these techniques focus on finding frequent subgraphs or subtrees. Statistical significance of the frequent patterns was not addressed.

Milo et al. [15] identified network motifs in complex networks. They defined network motifs as graph patterns that appear significantly more frequently than those in randomized networks. However, their method deals with a single large graph, whereas our model deals with a large collection of graphs. Moreover, they computed the p-value by simulation: they generated a number of randomized networks, and counted the number of networks that contained the subgraph with a support at least the observed support. This approach cannot compute p-values with high accuracy because the generation of N randomized networks can never yield a non-zero p-value of less than $1/N$. In contrast, our method is deterministic and computes accurate p-values.

In the study of large graphs such as the Internet, random graph models [19] are used to describe the graph topology. Faloutsos et al. [20] showed that degrees of nodes of the Internet follow a power-law distribution. Albert et al. [21] showed the small-world phenomena of the world-wide web. Leskovec et al. [22] observed how graphs evolve over time in terms of densities and diameters etc. Whereas these studies pertain to properties of the graph topology, we target the discovery of recurring subgraphs in a collection of graphs.

In an approach to interestingness measurement, Bayardo and Agrawal [23] proposed to mine an optimal set of rules according to a partial order defined using both rule support and confidence. Jaroszewicz and Simovici [24] defined the interestingness of frequent itemsets as the difference between the support from data and the support estimated from a background Bayesian network. Amer-Yahia et al. [25] proposed scoring methods based on both structure and content. The scoring methods are used for ranking answers to XML queries.

In the area of frequent itemset mining, Srikant and Agrawal [26] addressed the problem of mining quantitative association rules. Han et al. [27] developed an algorithm for mining top-K frequent closed patterns.

9 Conclusions

Statistical significance and ranking are useful in the post-processing of data mining results. In this paper, we proposed a probabilistic model for frequent subgraphs, and show that frequent subgraphs can be effectively ranked by their p-values. By representing graphs in the feature space, we derived a exact model which leads to a closed form solution for the p-values. Efficient methods were developed for computing the p-values and lower bounds. A simplified model was further proposed to improve efficiency. We also addressed the problem of feature vector mining, and developed an algorithm that efficiently searches significant closed sub-vectors. Experimental results validated the quality, performance, and practical usefulness of the presented techniques.

Although presented in the context of graphs, the proposed techniques are generic and can be applied to mining of other complex data, such as trees. Future directions are integration of the significance measurement and graph mining techniques, incorporation of feature dependency into the probabilistic model, and development of better approximations and lower bounds.

Acknowledgements: We would like to thank Xifeng Yan and Jiawei Han for providing the code of CloseGraph. The work was supported in part by NSF grants IIS-0612327 and DBI-0213903.

References

[1] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB J. 10(4): 334-350 (2001)*.

[2] S. Berretti, A. D. Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 23, 2001.

[3] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *KDD*, 2003.

[4] J. Hu, X. Shen, Y. Shao, C. Bystroff, and M. J. Zaki. Mining protein contact maps. In *BIOKDD*, 2002.

[5] R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. In *Proc Natl Acad Sci*, 2005.

[6] S. Kramer, L. D. Raedt, and C. Helma. Molecular feature mining in HIV data. In *KDD*, 2001.

[7] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures

from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000.

[8] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[9] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM*, 2002.

[10] X. Yan and J. Han. CloseGraph: Mining closed frequent graph patterns. In *KDD*, 2003.

[11] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *ICDM*, 2003.

[12] J. Huan, W. Wang, J. Prins, and J. Yang. SPIN: Mining maximal frequent subgraphs from graph databases. In *KDD*, 2004.

[13] N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data. In *Proceedings of ICDM*, 2002.

[14] N. Vanetik and E. Gudes. Mining frequent labeled and partially labeled graph patterns. In *ICDE*, 2004.

[15] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, October 2002.

[16] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*, chapter 5, pages 181–183. Academic press, second edition, 2003.

[17] M. J. Zaki. Efficiently mining frequent trees in a forest. In *KDD*, 2002.

[18] Y. Chi, Y. Yang, and R. R. Muntz. Indexing and mining free trees. In *ICDM*, 2003.

[19] P. Erdos and A. Renyi. On random graphs. In *Publ. Math.*, pages 290–297, 1959.

[20] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, 1999.

[21] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world-wide web. In *Nature*, pages 401:130–131, 1999.

[22] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *KDD*, 2005.

[23] R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In *KDD*, 1999.

[24] S. Jaroszewicz and D. A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In *KDD*, 2004.

[25] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman. Structure and content scoring for XML. In *VLDB*, 2005.

[26] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD*, 1996.

[27] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-K frequent closed patterns without minimum support. In *ICDM*, 2002.

APPENDIX

A Proof of Lemma 2

Proof. (1) According to Eqn. (6), let $\underline{x_1} = (x_i, x_j)$ and $\underline{x'_1} = (x_i + 1, x_j - 1)$. It is sufficient if we can show that $P(\underline{x_1}; s) \geq P(\underline{x'_1}; s)$ for all $s \geq x_i + x_j$.

$$
\begin{aligned}
P(\underline{x_1}; s) &= \sum_{t=x_i}^{s-x_j} \binom{s}{t} \theta_i^t \theta_j^{s-t} \\
P(\underline{x'_1}; s) &= \sum_{t=x_i+1}^{s-x_j+1} \binom{s}{t} \theta_i^t \theta_j^{s-t} \\
P(\underline{x_1}; s) &- P(\underline{x'_1}, s) \\
&= \binom{s}{x_i} \theta_i^{x_i} \theta_j^{s-x_i} - \binom{s}{x_j-1} \theta_i^{s-x_j+1} \theta_j^{x_j-1}
\end{aligned}
$$

Since $x_i \geq x_j$ and $s \geq x_i + x_j$, we get $\binom{s}{x_i} \geq \binom{s}{x_j-1}$. Since $\theta_i \leq \theta_j$ and $x_i \leq s - x_j + 1$, we get $\theta_i^{x_i} \theta_j^{s-x_i} \geq \theta_i^{s-x_j+1} \theta_j^{x_j-1}$. Thus, $P(\underline{x_1}; s) - P(\underline{x'_1}; s) \geq 0$.

(2) Let $\underline{x_1} = (x_i, x_j)$, $\underline{x'_1} = (x_j, x_i)$,

$$
\begin{aligned}
P(\underline{x_1}, s) &= \sum_{t=x_i}^{s-x_j} \binom{s}{t} \theta_i^t \theta_j^{s-t} \\
P(\underline{x'_1}, s) &= \sum_{t=x_j}^{s-x_i} \binom{s}{t} \theta_i^t \theta_j^{s-t}
\end{aligned}
$$

Since $x_i < x_j$ and $s \geq x_i + x_j$, let $b = min(s - x_j, x_j)$, then

$$
\begin{aligned}
&P(\underline{x_1}, s) - P(\underline{x'_1}, s) \\
&= \sum_{t=x_i}^{b} \binom{s}{t} \theta_i^t \theta_j^{s-t} - \sum_{t=s-b}^{s-x_i} \binom{s}{t} \theta_i^t \theta_j^{s-t} \\
&= \sum_{t=x_i}^{b} \binom{s}{t} \theta_i^t \theta_j^{s-t} - \sum_{t=x_i}^{b} \binom{s}{t} \theta_i^{s-t} \theta_j^t
\end{aligned}
$$

It can be verified that $t \leq s - t$ when $x_i \leq t \leq b$. It follows that $\theta_i^t \theta_j^{s-t} \geq \theta_i^{s-t} \theta_j^t$. Thus, $P(\underline{x_1}, s) - P(\underline{x'_1}, s) \geq 0$.

□