Efficient Discovery of Frequent Correlated Subgraph Pairs

Yiping Ke
The Chinese University of Hong Kong
ypke@se.cuhk.edu.hk

James Cheng
Nanyang Technological University
jamescheng@ntu.edu.sg

Jeffrey Xu Yu
The Chinese University of Hong Kong
yu@se.cuhk.edu.hk

***Abstract*—The recent proliferation of graph data in a wide spectrum of applications has led to an increasing demand for advanced data analysis techniques. In view of this, many graph mining techniques, such as *frequent subgraph* mining and *correlated subgraph* mining, have been proposed. In many applications, both frequency and correlation play an important role. Thus, this paper studies a new problem of mining the set of *frequent correlated subgraph pairs*. A simple algorithm that combines existing algorithms for mining frequent subgraphs and correlated subgraphs results in a multiplication of the mining operations, the majority of which are redundant. We discover that most of the graphs correlated to a common graph are also highly correlated. We establish theoretical foundations for this finding and derive a tight lower bound on the correlation of any two graphs that are correlated to a common graph. This theoretical result leads to the design of a very effective *skipping mechanism*, by which we skip the processing of a majority of graphs in the mining process. Our algorithm, *FCP-Miner*, is a fast approximate algorithm, but we show that the missing pairs are only a small set of marginally correlated pairs. Extensive experiments verify both the efficiency and effectiveness of FCP-Miner.**

***Keywords*-graph mining; Pearson's correlation coefficient; frequent correlated subgraph pairs**

I. Introduction

The use of graph-based representation has gained increasing popularity in various application domains, including bioinformatics [1], [2], chemistry [3], [4], drug design [5], [6], social network analysis [7], and many more. As a result, graph pattern mining has become an important research problem. Existing studies on graph pattern mining mainly focus on finding *frequent subgraphs* [8], [9], [10], [11], [12] and its closed or maximal variations [13], [14], [15], while little attention has been paid to finding other types of useful graph patterns.

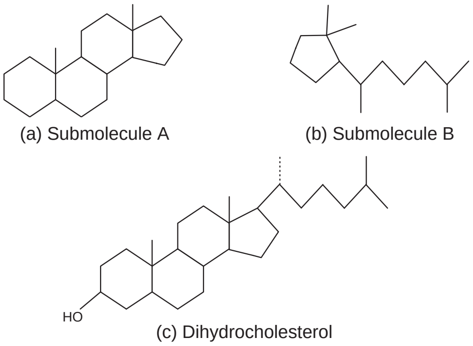
In traditional pattern mining, correlated patterns have also been recognized as an important type of patterns. Correlated pattern mining has been extensively studied in market-basket data [16], [17], [18], [19], [20], [21] and recently introduced to the context of graph data [22], [23]. A pair of subgraphs are correlated if their occurrence distributions are similar, which means that they are often co-present and co-absent, and thus have mutual implication on their occurrences. Given a *query graph*, existing work CGSearch [22] (or its top-k version [23]) returns all (or top-k) correlated graphs *with respect to the query graph*. Both of the studies assume the existence of an interesting subgraph to be served as the query graph. In practice, however, such apriori knowledge may not always be available.

In this paper, we study a new problem of discovering all frequent correlated subgraph pairs. Unlike the existing work, the new problem does not require the specification of a query graph. Instead, it aims to discover all correlations, which is a more general setting and more practical. The problem is formulated as follows: *Given a graph database $\mathcal{D}$ that contains a set of graphs, a minimum correlation threshold θ, and a minimum support threshold σ, find all pairs of frequent subgraphs whose correlation is at least θ.* The usage of σ allows a user to control the occurrence probability (also called *support*) of an interesting subgraph, while θ is used to specify how similar the occurrence distribution of a pair of correlated subgraphs is wanted. Thus, a frequent correlated subgraph pair is also called a *(σ, θ)-subgraph pair*. The correlation between two graphs is measured by a function of the individual support of the two graphs and their joint support. In this paper, we use the well-known Pearson's correlation coefficient ϕ [24] as the correlation measure.

The (σ, θ)-subgraph pairs are very useful in a wide range of applications. The following gives a concrete example in medicinal chemistry.

Example 1. Fig. 1(a) and (b) show a pair of correlated submolecules discovered from a real chemical compound structure database in the National Cancer Institute. Each vertex in the figure represents a carbon atom and each edge represents a single bond. Being a (σ, θ)-subgraph pair, it means that these two submolecules often accompany with each other in a specific set of compounds in the database. Interestingly, we find that these two submolecules represent a class of biologically active compounds, which is depicted in Fig. 1(c). This class contains Dihydrocholesterol analogues, which are cholesterol derivatives found in human feces, gallstones, eggs, and other biological matter. □

Example 1 shows several usages of (σ, θ)-subgraph pairs in medicinal chemistry. First, a (σ, θ)-subgraph pair is able to indicate the existence of a class of interesting compounds, of which may be unaware by the chemist. Therefore, it can direct the attention of the chemist to these hidden classes, which may lead to the discovery of new substances or



(a) Submolecule A

(b) Submolecule B

(c) Dihydrocholesterol

Figure 1. A pair of correlated submolecules and its represented compound class

drugs. Second, the (σ, θ)-subgraph pair captures the active structures of its represented compound class and thus can serve as building blocks (representative / functional submolecules) of the compound class, which is particularly useful in compound synthesis. Third, the set of all (σ, θ)-subgraph pairs can automatically form clusters of biologically active or well-investigated structures. This can be used as a summarization of the database so that the chemist can gain a biological insight of how the compounds are medically connected. Furthermore, the whole set of (σ, θ)-subgraph pairs can also guide chemists about their choices of submolecules when synthesizing new compounds. The co-occurrences of correlated submolecules indicate that they are relatively easy to synthesize. Therefore, chemists can be saved from a tremendous number of laboratorial tests by using correlated submolecules.

Being aware of the usefulness of graph correlations, researchers in medicinal chemistry have tried to discover correlated submolecules from compound databases [25]. However, the lack of efficient mining techniques hinders them from investigating more general structured submolecules, which inspires our work of discovering all (σ, θ)-subgraph pairs.

However, finding all (σ, θ)-subgraph pairs is a challenging problem. First, the number of frequent subgraphs in $\mathcal{D}$ can be large due to the high diversity in the structure of graph data. Second, for a specific subgraph f, every frequent subgraph is a candidate to form a (σ, θ)-subgraph pair with f. That is, the candidate set of f, $\mathcal{C}_f$, is equal to the set of all frequent subgraphs, $\mathcal{F}$. This results in an explosion in the number of candidates for subgraph pairs, which is $|\mathcal{F}| \times |\mathcal{F}|$. Third, unlike the support measure, the correlation measure does not have the anti-monotone property. That means if a graph g is found to be uncorrelated with f, we cannot prune all supergraphs of g as does in frequent subgraph mining, since they may still be correlated with f. As a result, the size of the candidate set, $|\mathcal{C}_f|$, cannot be effectively reduced. The lack of this powerful pruning property makes the design of an efficient algorithm very difficult.

There are two straightforward solutions to the problem based on the existing work of frequent subgraph mining [8], [9], [10], [11], [12] and query-based correlation mining [22].

The first one is a naive solution, which is a frequent-subgraph-mining-based approach. The idea is to first mine the set of all frequent subgraphs $\mathcal{F}$ using an existing mining algorithm and then check the correlation values among all pairs of frequent subgraphs. However, this approach has several drawbacks. First, mining all frequent subgraphs can be expensive. Second, checking pairwise correlations among all frequent subgraphs is usually infeasible since the number of all possible pairs can be prohibitively large, especially when σ is small. Furthermore, the joint support of each pair of subgraphs needs to be computed during correlation checking, which takes at least $(\sigma \cdot |\mathcal{D}|)$ number of intersections on the projected databases of the two subgraphs. Here, the projected database of a subgraph is defined to be the set of graphs in $\mathcal{D}$ that contain the subgraph. Therefore, the computational complexity of this naive approach is extremely high.

Another more feasible solution, which is a CGSearch-based approach, is to explore the subgraph space and feed each frequent subgraph into the query-based correlation mining algorithm, CGSearch [22]. This approach is inefficient as well since each invocation of CGSearch involves an expensive operation to mine the projected database of the query graph. Moreover, every correlated subgraph pair is computed twice by this approach since each of the subgraph in the pair is feeded as a query once.

In this paper, we propose an approximate but efficient solution to the problem. Given a frequent subgraph f, we define its *answer set*, denoted as $\mathcal{A}_f$, to be the set of subgraphs that form (σ, θ)-subgraph pairs with f. The main idea of our approach is to compute the exact answer sets of only a small number of frequent subgraphs by CGSearch and use these exact answer sets to approximate the answer sets of the remaining frequent subgraphs. In this way, our approach is able to skip invoking CGSearch for most of the frequent subgraphs, which saves tremendous computational costs compared with the above-mentioned CGSearch-based approach. Furthermore, our approach significantly reduces the size of the candidate sets from the whole set $\mathcal{F}$ to a much smaller set, which is far superior to the frequent-subgraph-mining-based approach.

The mechanism of skipping the processing of most frequent subgraphs has its theoretical foundations. We investigate the characteristics of correlated subgraphs and find that correlativeness tends to have the "*transitive*" property. More specifically, if two subgraphs f_1 and f_2 are found to be correlated with the same subgraph f, they are likely to be correlated as well. This observation is verified theoretically by deriving a tight lower bound of the correlation $\phi(f_1, f_2)$. The lower bound guarantees that an arbitrary pair of graphs f_1 and f_2 has a high correlation as long as they are both correlated with a third graph f, i.e., they appear in the answer set of the same graph f. Therefore, it is theoretically sound to approximate the answer sets of f_1 and f_2 using the answer set of f.

Based on the theoretical results, we develop an efficient algorithm to mine Frequent Correlated subgraph Pairs, namely *FCP-Miner*. The algorithm traverses the subgraph space in a depth-first manner. It processes a new subgraph f by CGSearch to obtain its exact answer set $\mathcal{A}_f$. Then, all graphs in $\mathcal{A}_f$ are marked to be "*skipped*". Thus, FCP-Miner processes only those new frequent subgraphs that are not marked as "*skipped*", while for each newly processed frequent subgraph FCP-Miner adds a bunch of graphs to the skip list. For each skipped graph $f_i \in \mathcal{A}_f$, FCP-Miner uses $\mathcal{A}_f$ as the candidate set of f_i. Its approximated answer set is then computed from this much smaller candidate set. In this way, FCP-Miner is able to skip most of the graphs from processing CGSearch and thus significantly improves the mining efficiency. However, it is possible for FCP-Miner to miss some subgraph pairs due to the approximation on the answer sets of the skipped graphs.

Our extensive experiments show that the number of missing pairs by FCP-Miner is very small. More importantly, we find that the correlation values of the missing pairs are close to θ, which means that they are just boundary pairs. This result indicates a high-quality of the approximation and the effectiveness of FCP-Miner. Furthermore, compared with the CGSearch-based approach that processes every frequent subgraph, FCP-Miner is over an order of magnitude faster because of the effective skipping mechanism. On average, about 80% of the frequent subgraphs can be skipped by FCP-Miner and the average candidate set size of the subgraphs is reduced by over 98%. This result demonstrates the efficiency of FCP-Miner.

The contributions of the paper are as follows.

- We propose a new problem of discovering all frequent correlated subgraph pairs from graph databases, which is important and demanding in a wide range of applications.
- We derive a tight lower bound for an arbitrary pair of subgraphs in the same answer set, which provides the theoretical guarantee for performing the high-quality approximation.
- We devise an efficient and approximate algorithm, FCP-Miner, which utilizes an effective skipping mechanism to reduce the search space.
- We conduct extensive experiments that verify the efficiency of our algorithm, as well as the high quality of the approximation.

The rest of the paper is organized as follows. Section II gives some preliminaries on graph database and frequent subgraphs. Section III defines the problem of frequent correlated subgraph pair discovery. Section IV presents our solution and its theoretical foundations. Section V evaluates the performance of our approach. Section VI reviews some related work. Finally, Section VII concludes the paper.

II. Preliminaries

Graphs studied in this paper are undirected, labeled and connected. A *graph* g is defined as a triple (V, E, l), where V is the set of vertices, E is the set of edges and l is a labeling function that assigns a label to each vertex and edge.

Given two graphs, $g = (V, E, l)$ and $g' = (V', E', l')$, g is called a *subgraph* of g' (or g' is a *supergraph* of g), denoted as $g \subseteq g'$ (or $g' \supseteq g$), if there exists an injective function f: $V \rightarrow V'$, such that $\forall (u, v) \in E$, $(f(u), f(v)) \in E'$, $l(u) = l'(f(u))$, $l(v) = l'(f(v))$, and $l(u, v) = l'(f(u), f(v))$. The injective function f is called a *subgraph isomorphism* from g to g'. Testing subgraph isomorphism between two graphs is known to be *NP-complete* [26].

A *graph database* $\mathcal{D}$ is a collection of graphs, denoted as $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$. Given $\mathcal{D}$ and a graph g, we define the *projected database* of g as the set of graphs in $\mathcal{D}$ that are supergraphs of g, denoted as $\mathcal{D}_g = \{g' : g' \in \mathcal{D}, \; g' \supseteq g\}$. The size of the projected database is called the *frequency* of g in $\mathcal{D}$, denoted as $\mathit{freq}(g) = |\mathcal{D}_g|$. The *support* of g in $\mathcal{D}$ is further defined as $supp(g) = \frac{\mathit{freq}(g)}{|\mathcal{D}|}$, which is the probability of a graph in $\mathcal{D}$ being the supergraph of g. A graph g is called a *Frequent subGraph* (*FG*) [8] in $\mathcal{D}$ if $supp(g) \geq \sigma$, where σ $(0 \leq \sigma \leq 1)$ is a user-specified *minimum support threshold*. We use $\mathcal{F}$ to denote the set of all FGs in $\mathcal{D}$ with respect to σ.

Given two graphs g and g', we define their *joint frequency* as the number of graphs in $\mathcal{D}$ that are common supergraphs of g and g', denoted as $\mathit{freq}(g, g') = |\mathcal{D}_g \cap \mathcal{D}_{g'}|$. The *joint support* of g and g' is defined as $supp(g, g') = \frac{\mathit{freq}(g,g')}{|\mathcal{D}|}$. The support measure is *anti-monotone*, that is, $g \subseteq g'$ implies that $supp(g) \geq supp(g')$. We also have the following properties of the joint support: $supp(g, g') \leq supp(g)$ and $supp(g, g') \leq supp(g')$.

III. Problem Definition

In this paper, we adopt *Pearson's correlation coefficient* [24] as the correlation measure, defined as follows.

Definition 1. (Pearson's Correlation Coefficient) *Given a pair of graphs g_1 and g_2, the* Pearson's correlation coefficient *of g_1 and g_2, denoted as $\phi(g_1, g_2)$, is defined as*

$$\phi(g_1, g_2) = \frac{supp(g_1, g_2) - supp(g_1)supp(g_2)}{\sqrt{supp(g_1)supp(g_2)(1-supp(g_1))(1-supp(g_2))}}. \quad (1)$$

When $supp(g_1)$ or $supp(g_2)$ is equal to 0 or 1, $\phi(g_1, g_2)$ is defined to be 0.

It is easy to see that ϕ is symmetric. The value of $\phi(g_1, g_2)$ is in the range of $[-1, 1]$. When $supp(g_1, g_2) = supp(g_1) \cdot supp(g_2)$, $\phi(g_1, g_2)$ takes the value of 0, which indicates that the occurrences of g_1 and g_2 are independent to each other.

The positive value of $\phi(g_1, g_2)$ indicates that the occurrences of g_1 and g_2 are positively correlated, i.e., they are often co-present and co-absent. On the other hand, the negative value shows negative correlation, i.e., g_1 often occurs without g_2, and vice versa. In this paper, we focus on finding positive correlations.

We now present two useful properties of the ϕ function, which can be proved easily by taking the derivative of ϕ.

Property 1. *If both $supp(g_1)$ and $supp(g_2)$ are fixed, $\phi(g_1, g_2)$ is monotonically increasing with $supp(g_1, g_2)$.*

Property 2. *If both $supp(g_1)$ and $supp(g_1, g_2)$ are fixed, $\phi(g_1, g_2)$ is monotonically decreasing with $supp(g_2)$.*

The problem of finding frequent correlated subgraph pairs can be formalized as follows.

Frequent Correlated Subgraph Pair Discovery Given a graph database $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$, a minimum correlation threshold θ ($0 \leq \theta \leq 1$), and a minimum support threshold σ ($0 \leq \sigma \leq 1$), find all pairs of subgraphs f_1 and f_2 in $\mathcal{D}$ such that $supp(f_1) \geq \sigma$, $supp(f_2) \geq \sigma$, and $\phi(f_1, f_2) \geq \theta$.

We call such a pair of subgraphs a (σ, θ)-subgraph pair. Given a frequent subgraph f, we call the set of subgraphs that form (σ, θ)-subgraph pairs with f the *answer set* of f, denoted as $\mathcal{A}_f = \{f'\colon supp(f') \geq \sigma,\ \phi(f, f') \geq \theta\}$. Thus, the problem of frequent correlated subgraph pair discovery is essentially to find the $\mathcal{A}_f$ for each frequent subgraph f. We also use $\mathcal{C}_f$ to denote the set of candidate graphs that may potentially form (σ, θ)-subgraph pairs with f.

IV. Our Solution

In this section, we first derive the lower bound of the correlation among all subgraph pairs in the same answer set and then present our mining algorithm, FCP-Miner.

A. Correlation Lower Bound of Subgraph Pairs

Given a frequent subgraph f, we aim to derive a correlation lower bound of an arbitrary pair of graphs f_1 and f_2 in $\mathcal{A}_f$, denoted as $lower_{\phi(f_1, f_2)}$. In order to be computable and useful, the lower bound should be a function of the two known variables, $supp(f)$ and the minimum correlation threshold θ. For clarity of presentation, we let $a = supp(f)$.

We first review two useful lemmas proposed in CGSearch [22]. One lemma gives the bounds on the support of a graph $f' \in \mathcal{A}_f$, while the other gives the bounds on the joint support of f and $f' \in \mathcal{A}_f$.

Lemma 1. *Given a graph $f' \in \mathcal{A}_f$, the following lower and upper bounds of $supp(f')$, denoted respectively as $lower_{supp(f')}$ and $upper_{supp(f')}$, hold:*

$$lower_{supp(f')} = \frac{a}{\theta^{-2}(1-a) + a}, \tag{2}$$

$$upper_{supp(f')} = \frac{a}{\theta^{2}(1-a) + a}. \tag{3}$$

Lemma 2. *Given a graph $f' \in \mathcal{A}_f$, the following lower and upper bounds of the joint support $supp(f, f')$, denoted respectively as $lower_{supp(f,f')}$ and $upper_{supp(f,f')}$, hold:*

$$lower_{supp(f,f')} = \frac{a}{\theta^{-2}(1-a) + a}, \tag{4}$$

$$upper_{supp(f,f')} = a. \tag{5}$$

The above two lemmas specify only the bounds with respect to a single graph $f' \in \mathcal{A}_f$, while the relationship between two graphs in $\mathcal{A}_f$ is still lacking, which is exactly what we try to get.

Given an arbitrary pair of graphs f_1 and f_2 in $\mathcal{A}_f$, by the definition of ϕ, we need to know the values of their individual support $supp(f_1)$, $supp(f_2)$, and their joint support $supp(f_1, f_2)$, in order to obtain the value of $\phi(f_1, f_2)$. According to Property 1, $\phi(f_1, f_2)$ is monotonically increasing with $supp(f_1, f_2)$. Therefore, in order to get the lower bound of $\phi(f_1, f_2)$, we first derive a lower bound of $supp(f_1, f_2)$, which is stated in the following theorem.

Theorem 1. *Given a graph f and two graphs f_1, f_2 in $\mathcal{A}_f$, the following lower bound of $supp(f_1, f_2)$ holds:*

$$supp(f_1, f_2) \geq supp(f, f_1) + supp(f, f_2) - supp(f). \tag{6}$$

The equality holds when $(\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}) \subseteq \mathcal{D}_f \subseteq (\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2})$.

Proof: By the definitions of the support and the joint support, (6) holds if and only if the following inequality holds.

$$|\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}| \geq |\mathcal{D}_f \cap \mathcal{D}_{f_1}| + |\mathcal{D}_f \cap \mathcal{D}_{f_2}| - |\mathcal{D}_f|. \tag{7}$$

To prove (7), we decompose $|\mathcal{D}_f|$ as follows.

$$\begin{aligned}
|\mathcal{D}_f| &\geq |\mathcal{D}_f \cap (\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2})| & (8)\\
&= |(\mathcal{D}_f \cap \mathcal{D}_{f_1}) \cup (\mathcal{D}_f \cap \mathcal{D}_{f_2})| & (9)\\
&= |\mathcal{D}_f \cap \mathcal{D}_{f_1}| + |\mathcal{D}_f \cap \mathcal{D}_{f_2}| - |\mathcal{D}_f \cap \mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}| & (10)\\
&\geq |\mathcal{D}_f \cap \mathcal{D}_{f_1}| + |\mathcal{D}_f \cap \mathcal{D}_{f_2}| - |\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}|. & (11)
\end{aligned}$$

Equation (8) holds since the intersection of $\mathcal{D}_f$ with any set is a subset of $\mathcal{D}_f$. Equation (9) is by the distributive law of the set intersection over the set union. Equation (10) is by the inclusion-exclusion principle. Equation (11) holds since $(\mathcal{D}_f \cap \mathcal{D}_{f_1} \cap \mathcal{D}_{f_2})$ is a subset of $(\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2})$, by which (7) follows.

The equality holds when $(\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2}) \supseteq \mathcal{D}_f$ for (8) and $\mathcal{D}_f \supseteq (\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2})$ for (11). Therefore, the equality holds when $(\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}) \subseteq \mathcal{D}_f \subseteq (\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2})$. ■

Theorem 1 serves as a bridge between $supp(f_1, f_2)$ and the two joint support $supp(f, f_1)$ and $supp(f, f_2)$, whose bounds are known in Lemma 2.

By Property 1 and the definition of ϕ, we can replace $supp(f_1, f_2)$ with the results in Theorem 1 into $\phi(f_1, f_2)$ and obtain the following inequality.

$$\phi(f_1, f_2) \geq \frac{supp(f,f_1)+supp(f,f_2)-a-supp(f_1)supp(f_2)}{\sqrt{supp(f_1)supp(f_2)(1-supp(f_1))(1-supp(f_2))}}. \tag{12}$$

Although (12) already specifies a lower bound of $\phi(f_1, f_2)$, it is not a useful one since all the variables except a in this lower bound are unknown. Therefore, we need to make further derivations.

The right side of (12) is monotonically increasing with $supp(f, f_1)$ and $supp(f, f_2)$ and is monotonically decreasing with $supp(f_1)$ and $supp(f_2)$ if other variables are fixed. Therefore, we can simply replace these variables in (12) respectively with $lower_{supp(f,f_1)}$, $lower_{supp(f,f_2)}$, $upper_{supp(f_1)}$ and $upper_{supp(f_2)}$, which are given in Lemmas 1 and 2. However, the lower bound of $\phi(f_1, f_2)$ derived in this way is not tight. This is because the joint support $supp(f, f_1)$ is dependent on the individual support $supp(f_1)$. As a result, $lower_{supp(f,f_1)}$ and $upper_{supp(f_1)}$ cannot be achieved simultaneously. The case is the same for $lower_{supp(f,f_2)}$ and $upper_{supp(f_2)}$. Thus, the lower bound of $\phi(f_1, f_2)$ derived in this simple way is a loose one and has no guarantee.

Based on the above analysis, we need to resolve the dependency of the joint support and the individual support in (12) so as to obtain a tight bound. This can be accomplished as follows. Recall that both f_1 and f_2 are in $\mathcal{A}_f$, which implies that $\phi(f, f_1) \geq \theta$ and $\phi(f, f_2) \geq \theta$. By the definition of ϕ, we have the following inequality:

$$\begin{aligned} supp(f, f_1) \geq {} & \theta\sqrt{a(1-a)supp(f_1)(1-supp(f_1))} \\ & + a \cdot supp(f_1). \end{aligned} \tag{13}$$

The right hand side of (13) is a function of $supp(f_1)$, which can be used to resolve the dependency problem. The equality holds when $\phi(f, f_1) = \theta$. We have similar results for $supp(f, f_2)$. By replacing $supp(f, f_1)$ and $supp(f, f_2)$ with (13) in (12), we obtain the following inequality of $\phi(f_1, f_2)$. For clarity, we let $x = supp(f_1)$ and $y = supp(f_2)$.

$$\begin{aligned} \phi(f_1, f_2) \geq {} & \frac{\theta\sqrt{a(1-a)}(\sqrt{x(1-x)}+\sqrt{y(1-y)})+a(x+y)-a-xy}{\sqrt{x(1-x)y(1-y)}} \\ = {} & h(x, y). \end{aligned} \tag{14}$$

The equality holds when $(\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}) \subseteq \mathcal{D}_f \subseteq (\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2})$ and $\phi(f, f_1) = \phi(f, f_2) = \theta$.

The right hand side of (14) is a function of x and y, denoted as $h(x, y)$. More importantly, the variables x and y can take independent values in the function h. In order to utilize the bounds in Lemma 1 to obtain a tight lower bound of $\phi(f_1, f_2)$, we need to know the monotonicity of x and y in $h(x, y)$. Since x and y are symmetric in $h(x, y)$, we only consider the monotonicity of x, which is given in the following lemma.

Lemma 3. *Let $t = \sqrt{\frac{a(1-y)}{(1-a)y}}$. The following two statements about the monotonicity of x in $h(x, y)$ are true:*

1. *The function h is monotonically increasing with x in $[lower_{supp(f_1)}, \frac{t-\theta}{t+t^{-1}-2\theta}]$;*
2. *The function h is monotonically decreasing with x in $[\frac{t-\theta}{t+t^{-1}-2\theta}, upper_{supp(f_1)}]$.*

Proof: In order to determine the monotonicity of x, we apply the differentiation to the function h with respect to x and obtain the following equality.

$$h'(x) = \frac{(a-y)+\sqrt{a(1-y)}(\theta\sqrt{(1-a)y}-\sqrt{a(1-y)})(2-x^{-1})}{2\sqrt{y(1-y)x}(\sqrt{1-x})^3}. \tag{15}$$

The sign of $h'(x)$ is the same as the sign of the numerator in (15), denoted as

$$M = (a-y)+\sqrt{a(1-y)}(\theta\sqrt{(1-a)y}-\sqrt{a(1-y)})(2-x^{-1}).$$

We first show that $(\theta\sqrt{(1-a)y} - \sqrt{a(1-y)}) \leq 0$. It is easy to see that the left hand side of this inequality monotonically increases with y. The inequality follows when we replace y with its upper bound $upper_{supp(f_2)}$ as given in Lemma 1.

In order to have $M \geq 0$, it is equivalent to have the following inequalities.

$$\begin{aligned} & M \geq 0 \\ \Leftrightarrow \quad & \sqrt{a(1-y)}(\theta\sqrt{(1-a)y}-\sqrt{a(1-y)})(2-x^{-1}) \geq (y-a) \\ \Leftrightarrow \quad & x^{-1} \geq 2 + \frac{y-a}{a(1-y)-\theta\sqrt{a(1-a)y(1-y)}} \\ & \quad = 1 + \frac{(1-a)y-\theta\sqrt{a(1-a)y(1-y)}}{a(1-y)-\theta\sqrt{a(1-a)y(1-y)}} \\ & \quad = 1 + \frac{t^{-1}-\theta}{t-\theta}. \end{aligned} \tag{16}$$

By taking the derivative of the function $t = \sqrt{\frac{a(1-y)}{(1-a)y}}$, we find that t is monotonically decreasing with y. By Lemma 1, we have $lower_{supp(f_2)} \leq y \leq upper_{supp(f_2)}$. By replacing the bounds of y in t, we have $\theta \leq t \leq \theta^{-1}$. Therefore, (16) is always positive.

By (16), it follows that $M \geq 0$ or equivalently $h'(x) \geq 0$ if and only if

$$x \leq \frac{t-\theta}{t+t^{-1}-2\theta}.$$

Following the similar derivation, we can get that $M \leq 0$ or equivalently $h'(x) \leq 0$ if and only if

$$x \geq \frac{t-\theta}{t+t^{-1}-2\theta}.$$

By Lemma 1, we have $lower_{supp(f_1)} \leq x \leq upper_{supp(f_1)}$, the monotonicity of x in h thus follows. ■

According to Lemma 3, when y is fixed, the minimum value of $h(x,y)$ can be achieved at either $x = lower_{supp(f_1)}$ or $x = upper_{supp(f_1)}$. Since the variable y is independent of x, the minimum value of $h(x,y)$ can only occur at four points, which are combinations of x and y taking their lower and upper bounds. Since x and y are symmetric, there are essentially three possible points. We summarize the lower bound of $\phi(f_1, f_2)$ in the following theorem.

Theorem 2. *Given a graph f and two graphs f_1, f_2 in its answer set $\mathcal{A}_f$, the following two statements of the lower bound of $\phi(f_1, f_2)$, denoted as $lower_{\phi(f_1,f_2)}$, hold:*

(1) *If $a \geq \frac{1}{2}$, then*

$$lower_{\phi(f_1,f_2)} = \theta^2 - (\theta^{-1} - \theta)^2 a,$$

which is achieved when $supp(f_1) = supp(f_2) = upper_{supp(f_1)}$ and $supp(f_1, f_2) = a$.

(2) *If $a < \frac{1}{2}$, then*

$$lower_{\phi(f_1,f_2)} = \theta^2 - (\theta^{-1} - \theta)^2 (1-a),$$

which is achieved when $supp(f_1) = supp(f_2) = lower_{supp(f_1)}$ and $supp(f_1, f_2) = 2 \cdot lower_{supp(f_1)} - a$.

Proof: We first check the minimum value of the function $h(x,y)$, which can only be achieved in either of the following three cases as discussed above.

When $x = lower_{supp(f_1)}$ and $y = lower_{supp(f_2)}$, by Lemma 1 and (14), we have

$$h(x,y) = \theta^2 - (\theta^{-1} - \theta)^2 (1-a). \quad (17)$$

When $x = lower_{supp(f_1)}$ and $y = upper_{supp(f_2)}$,

$$h(x,y) = \theta^2. \quad (18)$$

When $x = upper_{supp(f_1)}$ and $y = upper_{supp(f_2)}$,

$$h(x,y) = \theta^2 - (\theta^{-1} - \theta)^2 a. \quad (19)$$

Since the second term $(\theta^{-1} - \theta)^2(1-a)$ in (17) is non-negative, we have $(\theta^2 - (\theta^{-1} - \theta)^2(1-a)) \leq \theta^2$, which indicates that (18) does not specify a minimum value of h.

We now consider (17) and (19) and have the following derivations.

$$\begin{aligned} & \theta^2 - (\theta^{-1} - \theta)^2 (1-a) \ \geq \ \theta^2 - (\theta^{-1} - \theta)^2 a \\ \Leftrightarrow \quad & 1 - a \ \leq \ a \\ \Leftrightarrow \quad & a \ \geq \ \tfrac{1}{2}. \end{aligned}$$

Thus, when $a \geq \frac{1}{2}$, (19) serves as the lower bound of h. By (14), $\phi(f_1, f_2)$ is equal to h if and only if $(\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}) \subseteq \mathcal{D}_f \subseteq (\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2})$ and $\phi(f, f_1) = \phi(f, f_2) = \theta$. The condition of (19), i.e., $x = y = upper_{supp(f_1)}$, implies that $\phi(f, f_1) = \phi(f, f_2) = \theta$ holds. By replacing the values of x and y in (6), we have $supp(f_1, f_2) = a$, which ensures that $(\mathcal{D}_{f_1} \cap \mathcal{D}_{f_2}) \subseteq \mathcal{D}_f \subseteq (\mathcal{D}_{f_1} \cup \mathcal{D}_{f_2})$ holds.

Accordingly, when $a < \frac{1}{2}$, (17) is the lower bound of $\phi(f_1, f_2)$. By (6), this lower bound is achieved when $x = y = lower_{supp(f_1)}$ and $supp(f_1, f_2) = 2 \cdot lower_{supp(f_1)} - a$. ■

Theorem 2 provides a tight lower bound of the correlation between two arbitrary graphs f_1 and f_2 in the same answer set $\mathcal{A}_f$ of a graph f. It indicates that all the graphs in $\mathcal{A}_f$ have a correlation guarantee, which is not very far away from the threshold θ. For example, if $\theta = 0.9$ and $a = 0.5$, the lower bound computed by Theorem 2 is 0.79, which is still pretty high.

B. Our Algorithm

We first describe an exact algorithm that utilizes CGSearch for all FGs to mine the set of all (σ, θ)-subgraph pairs. Then, we make use of the theoretical results we obtain in the previous section to design an efficient approximate algorithm.

The exact algorithm has the following two components: (1) *FG-Enumerator*: the algorithm for enumerating all FGs; and (2) *CGSearch* [22]: an existing algorithm for mining the set of all correlated subgraphs of a given query graph f. The exact algorithm operates as follows. For each FG f returned by FG-Enumerator, find the set of all correlated subgraphs of f using CGSearch, and return those correlated subgraphs that have support at least σ.

The FG-Enumerator in the exact algorithm is used to enumerate the subgraphs in the pattern space in a depth-first manner similar to an FG mining algorithm such as gSpan [11]. However, it is different from an FG mining algorithm in the sense that it does not count the support of every FG, which is a very costly operation since it involves many subgraph isomorphism tests. FG-Enumerator works with CGSearch closely as follows. When the correlated subgraphs of f are returned by CGSearch, their support is obtained as well. Therefore, FG-enumerator does not need

Algorithm 1 *FCP-Miner*

```
Input: $\mathcal{D}$, $\sigma$ and $\theta$.
Output: A set of $(\sigma, \theta)$-subgraph pairs.

1. Initialize an empty hashtable $H$;
2. Invoke FG-Enumerator (with input $\sigma$ and $\mathcal{D}$);
3. for each FG $f$ returned by FG-Enumerator do
4.   if($f$ is not in $H$)
5.       Invoke CGSearch (with input $f$, $\theta$ and $\mathcal{D}$)
         to mine the set of correlated subgraphs of $f$, $\mathcal{C}_f$;
6.       $\mathcal{A}_f \leftarrow \{f_i : f_i \in \mathcal{C}_f, supp(f_i) \geq \sigma\}$;
7.       Output $f$ and $\mathcal{A}_f$;
8.       for each $f_i \in \mathcal{A}_f$ do
9.           $\mathcal{C}_{f_i} \leftarrow \mathcal{A}_f$;
10.          if($f_i$ is not in $H$)
11.              $\mathcal{A}_{f_i} \leftarrow \{f\} \cup \{f_j : f_j \in \mathcal{C}_{f_i}, \phi(f_i, f_j) \geq \theta, i \neq j\}$;
12.              Push $f_i$ and $\mathcal{A}_{f_i}$ into $H$;
13.          else            /* $f_i$ is in $H$ */
14.              Update $\mathcal{A}_{f_i} \leftarrow \mathcal{A}_{f_i} \cup$
                        $\{f_j : f_j \in \mathcal{C}_{f_i}, \phi(f_i, f_j) \geq \theta, i \neq j\}$;
15. Output $f$ and $\mathcal{A}_f$ for all $f$ in $H$;
```

to re-calculate the support of these subgraphs when they are explored later. Thus, FG-enumerator is more efficient than an FG mining algorithm to be incorporated with CGSearch for the purpose of mining (σ, θ)-subgraph pairs.

Since pairwise correlation is symmetric, the exact algorithm at least processes each same (σ, θ)-subgraph pair twice. In fact, a careful investigation will find that such redundant processing extends to sets of mutually correlated FGs. For example, if k FGs are mutually correlated, the exact algorithm needs to process $k(k-1)$ pairs. If duplicate pairs are processed only once, then we only process $k(k-1)/2$ pairs, but we still need to invoke CGSearch k times, which is still very expensive.

According to our finding in Theorem 2, given an FG f and the set of its correlated subgraphs $\mathcal{A}_f$ computed by CGSearch, the correlated subgraphs in $\mathcal{A}_f$ have high correlation with each other. Thus, we can utilize the answer set $\mathcal{A}_f$ of a discovered FG f as the candidate sets of all the subgraphs in $\mathcal{A}_f$. That is, we make $\mathcal{C}_{f_i} = \mathcal{A}_f$, $\forall f_i \in \mathcal{A}_f$. In this way, we reduce the candidate sets of many subgraphs from the whole set of all FGs $\mathcal{F}$ to a much smaller set $\mathcal{A}_f$.

We describe our algorithm, *FCP-Miner*, as follows.

FCP-Miner operates in a similar way to the exact algorithm except that we apply a *skipping* mechanism in Line 4 of Algorithm 1. That is, for all FGs returned by FG-Enumerator, we skip those FGs that already appear in the answer set of some previously processed FGs. These skipped FGs are kept in a hashtable once they are discovered to be correlated with some FG during the mining process (Lines 10-12), and they are skipped in Line 4 by checking the hashtable. The FGs are hashed into the hashtable by their canonical label (we use the minimum DFS code in gSpan [11]).

In other words, in Algorithm 1, we apply a heuristic implied by Theorem 2 which approximates the answer set of a skipped FG by the union of all its correlated subgraphs found in the answer sets of those processed FGs (Line 14). Therefore, if the FGs in the answer set of an FG indeed share high mutual correlation, then a huge amount of redundant processing can be avoided. For example, in the example of the k mutually correlated FGs mentioned earlier, we only need to invoke CGSearch once to obtain all the (σ, θ)-subgraph pairs among these k subgraphs.

Complexity Analysis. We analyze the complexity of FCP-Miner with comparison to the exact algorithm. For the exact algorithm, the complexity is $\mathcal{O}(h_1(\mathcal{F}) + |\mathcal{F}|h_2(n))$, where $h_1(\mathcal{F})$ is the complexity of exploring the set of FGs $\mathcal{F}$ using FG-Enumerator, and $h_2(n)$ is the complexity for obtaining n correlated subgraphs using CGSearch, assuming that n is the average number of correlated subgraphs of an FG in $\mathcal{F}$.

For FCP-Miner, due to the use of the skipping mechanism, the complexity is reduced to $\mathcal{O}(h_1(\mathcal{F}) + mh_2(n))$, where m is the number of those FGs that are not skipped. In general, m is significantly smaller than $|\mathcal{F}|$ and therefore tremendous saving is obtained, since the dominant factor in the complexity of the exact algorithm is $|\mathcal{F}|h_2(n)$. As we show in the following section, m is only a very small percentage of $|\mathcal{F}|$ in our experiments.

V. Performance Evaluation

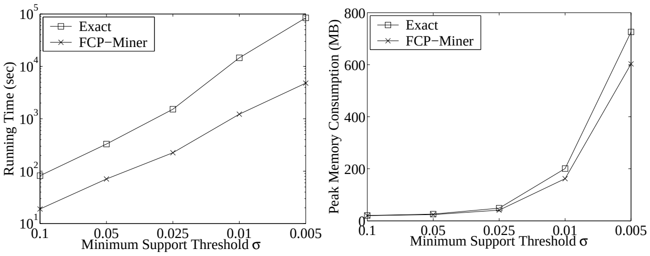
We evaluate the performance of our FCP-Miner algorithm on various metrics. The algorithm was implemented in C++. We ran all experiments on an AMD Opteron 248 with 8GB RAM, running Linux 64-bit.

Baseline of Comparison. Since this is the first proposal to mine (σ, θ)-subgraph pairs, we compare with the exact algorithm described in Section IV-B, denoted by *Exact* in our experiments. By comparing with Exact, we can demonstrate the effectiveness of the skipping mechanism in FCP-Miner as well as the approximation quality of FCP-Miner. We note that we do not compare with the frequent-subgraph-mining-based approach described in Section I because its complexity is $\mathcal{O}(h_3(\mathcal{F}) + |\mathcal{F}|^2\sigma|\mathcal{D}|)$, where $h_3(\mathcal{F})$ is the complexity of mining $\mathcal{F}$ using an FG mining algorithm. This computational complexity is higher than that of *Exact* since FG-Enumerator is faster than an FG mining algorithm for the purpose of pattern enumeration in mining (σ, θ)-subgraph pairs, and n is significantly smaller than $|\mathcal{F}|$. The frequent-subgraph-mining-based algorithm runs extremely slow for some of the datasets we test, especially when $|\mathcal{F}|$ is large and/or $|\mathcal{D}|$ is large.

Graph Databases. We test two real datasets most popularly used in the literature for evaluating FG and correlated-subgraph mining algorithms: *AIDS* and *NCI*. *AIDS* is the *AIDS antiviral screen dataset*, which contains 10K graphs.

Table I
CHARACTERISTICS OF DATASETS

	Range of graph size	Average graph size	Range of density	Average density
AIDS	1 − 217	27.40	0.009 − 1.0	0.10
NCI	1 − 252	19.95	0.008 − 1.0	0.14



(a) Running time (b) Peak memory consumption

Figure 2. Performance on varying σ

The *NCI* dataset contains 100K graphs, which we obtain from the National Cancer Institute database. Table I lists some characteristics of the datasets and more details can be found in their webpages[1].

A. Effect of σ

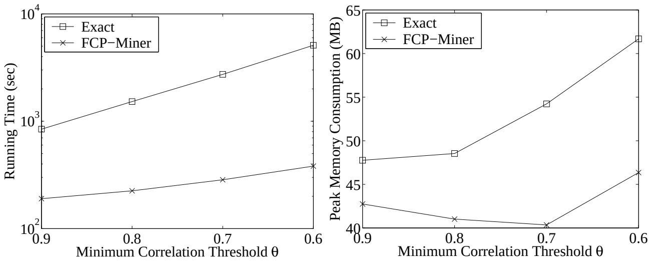
For the two parameters in a (σ, θ)-subgraph pair, we first test the effect of the first parameter σ on the performance of FCP-Miner. We decrease σ from 0.1 down to 0.005, by fixing θ at 0.8. We use the AIDS dataset for this experiment.

Fig. 2 reports the total running time and the peak memory consumption of mining (σ, θ)-subgraph pairs using FCP-Miner and Exact, respectively. The result clearly shows that FCP-Miner is significantly faster than Exact and more scalable as we lower the value of σ. FCP-Miner is almost an order of magnitude faster than Exact at a moderate σ of 0.025 and the improvement increases to 17.6 times when $\sigma = 0.005$. The memory consumption of the two algorithms is comparable, which is proportional to the number of FGs mined.

We further analyze the effectiveness of our approach by recording the following experimental findings, as reported in Table II: (1) *% of skip*: the percentage of FGs skipped by the skipping mechanism in FCP-Miner, calculated as $\frac{\text{total number of FGs skipped}}{\text{total number of FGs}}$; (2) *% of miss*: the percentage of (σ, θ)-subgraph pairs not found by FCP-Miner, calculated as $(1 - \frac{\text{total number of pairs obtained by FCP-Miner}}{\text{total number of pairs obtained by Exact}})$; (3) *avg ϕ of miss*: the average ϕ value of the missing pairs; (4) *avg ϕ of all*: the average ϕ value of all (σ, θ)-subgraph pairs; (5) *cand reduced*: the percentage of the candidates reduced by FCP-Miner from the whole set of FGs, computed as $(1 - \frac{\text{average number of candidates in FCP-Miner}}{\text{total number of FGs}})$; (6) *avg ans size*: the average size of the answer set of an FG.

Table II
THE EFFECTIVENESS OF FCP-MINER AT VARYING σ (FROM 0.1 TO 0.005)

	0.1	0.05	0.025	0.01	0.005
% of skip	67%	72%	78%	82%	85%
% of miss	5.9%	4.4%	3.0%	4.4%	4.4%
avg ϕ of miss	0.81	0.82	0.82	0.82	0.82
avg ϕ of all	0.91	0.91	0.91	0.91	0.90
cand reduced	98.65%	99.35%	99.75%	99.92%	99.96%
avg ans size	7.63	9.73	18.10	54.75	119.56



(a) Running time (b) Peak memory consumption

Figure 3. Performance on varying θ

We now analyze the effectiveness of FCP-Miner by the results reported in Table II. First, the performance improvement shown in Fig. 2(a) can be explained by *% of skip*, which shows that on average about 80% of the FGs are skipped by the skipping mechanism of FCP-Miner. It also shows that the percentage of FGs being skipped is higher for smaller σ, which is due to the fact that the average size of the answer set of an FG, *avg ans size*, is also larger for smaller σ. In addition, *cand reduced* indicates that FCP-Miner is very effective in reducing the correlation search space. Over 98% of the FGs are pruned from the candidate set so that a large number of correlation checking are successfully avoided.

Although the percentage of FGs being skipped is high, *% of miss* shows that the percentage of (σ, θ)-subgraph pairs missed by FCP-Miner is small, which is about 4% in most cases. Importantly, *avg ϕ of miss* shows that the missing (σ, θ)-subgraph pairs have the lowest ϕ values, which is close to $\theta = 0.8$, while *avg ϕ* shows that the average ϕ value of all (σ, θ)-subgraph pairs is much higher than θ.

In summary, these results reveal that FCP-Miner is efficient since most of the FGs are skipped, and effective since the percentage of missing (σ, θ)-subgraph pairs is small and the missing pairs are barely correlated with respect to the minimum correlation threshold θ.

B. Effect of θ

We now test the effect of the second parameter in a (σ, θ)-subgraph pair, θ, on the performance of FCP-Miner. We lower θ from 0.9 down to 0.6, by fixing σ at 0.025. We use the AIDS dataset for this experiment.

Fig. 3 reports the total running time and the peak memory consumption of FCP-Miner and Exact for the different

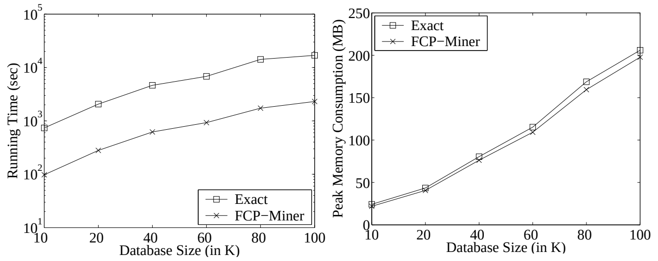
[1] *AIDS*: http://dtp.nci.nih.gov/
NCI: http://cactus.nci.nih. gov/ncidb2/download.html

Table III
THE EFFECTIVENESS OF FCP-MINER AT VARYING θ (FROM 0.9 TO 0.6)

	0.9	0.8	0.7	0.6
% of skip	65%	78%	84%	89%
% of miss	1.6%	3.0%	5.6%	4.9%
avg ϕ of miss	0.91	0.82	0.73	0.65
avg ϕ	0.95	0.91	0.86	0.80
cand reduced	99.87%	99.75%	99.57%	99.25%
avg ans size	11.02	18.10	26.74	38.17

Table IV
THE EFFECTIVENESS OF FCP-MINER AT VARYING DATABASE SIZES (FROM 10K TO 100K)

	10K	20K	40K	60K	80K	100K
% of skip	81%	79%	79%	79%	79%	78%
% of miss	2.5%	2.1%	1.7%	2.1%	3.3%	4.9%
avg ϕ of miss	0.82	0.82	0.82	0.82	0.82	0.82
avg ϕ	0.93	0.92	0.92	0.92	0.90	0.90
cand reduced	99.29%	99.38%	99.42%	99.43%	99.35%	99.41%
avg ans size	18.66	19.49	19.11	19.27	22.62	19.69



(a) Running time

(b) Peak memory consumption

Figure 4. Performance on different database sizes

values of θ. Again, the result shows that FCP-Miner is significantly faster than Exact, and also more scalable to the change of θ, especially when θ is smaller. The improvement of FCP-Miner over Exact is about an order of magnitude faster at $\theta = 0.8$, and the difference doubles when θ is lowered to 0.6. The memory consumption of FCP-Miner is also lower.

The effectiveness of FCP-Miner is assessed in Table III. When θ is high, the number of correlated subgraphs of each FG is small, as confirmed by *avg ans size*. Since FCP-Miner skips those FGs that appear in the answer set of the non-skipped FGs, the percentage of skipped FGs is also relatively smaller at higher values of θ, as shown by *% of skip*. However, this does not indicate that FCP-Miner is not effective for high values of θ. In fact, the percentage of candidates reduced by FCP-Miner is high for a high θ of 0.9.

Table III also verifies that the percentage of the missing pairs remains to be low for all θ values. The missing (σ, θ)-subgraph pairs are minimally correlated since their ϕ values are merely above the minimum correlation threshold θ. Note that the average ϕ value of all (σ, θ)-subgraph pairs is much greater than the respective θ.

C. Effect of Database Size

We also test the effect of database size on the performance of FCP-Miner. We create six NCI datasets, with sizes ranging from 10K to 100K graphs. The values of σ and θ are fixed at 0.05 and 0.8, respectively.

Fig. 4 shows that both the running time and the peak memory consumption of FCP-Miner and Exact increase linearly when the size of the database increases. The result shows that in all cases, FCP-Miner is almost an order of magnitude faster than Exact, though the memory consumption is comparable. The speed-up ratio of FCP-Miner over Exact remains rather stable when the database size increases, which is due to the fact that the number of all (σ, θ)-subgraph pairs does not vary significantly for different database sizes (this is also true with the number of FGs). This is further confirmed by the stable values of *avg ans size* over the different database sizes as shown in Table IV.

The results in Table IV show that FCP-Miner is able to skip most of the FGs during the mining process, while keeping a low percentage of missing (σ, θ)-subgraph pairs. The ϕ values of the missing pairs are also low as compared with the average ϕ value of all pairs. Therefore, we conclude that FCP-Miner is also both efficient and effective for mining (σ, θ)-subgraph pairs with varying database sizes.

VI. RELATED WORK

In the literature, correlated pattern mining has been widely studied in various types of databases. For market-basket data, a correlated pattern [16], [17], [18], [19], [20], [21] is composed of two or more basket items. Efficient algorithms were proposed to discover all correlated itemsets defined by various correlation measures such as the χ^2 test [16], [17], the m-pattern measure [18], h-confidence [19], Pearson's correlation coefficient [20], [21], etc. For stream data [27], [28], lagged correlation based on Pearson's correlation coefficient was proposed to study the lead-lag relationship between two time series. For multimedia data [29], correlation between multimedia objects was investigated for the task of automatic captioning of media data. Our work is incomparable to these studies due to different natures of data types.

In the context of graph databases, there are only three existing studies on correlation discovery, the stepwise correlated pattern mining [30], CGSearch [22], and the top-k version of CGSearch [23]. The work of [30] discovered top-k subgraphs that are correlated to a class attribute and used them as features for graph classification. CGSearch and its top-k version studied the correlation between a subgraph and a given query graph. Our work, on the other hand, does not require the user to specify a class attribute or a query graph but aims to discover all frequent correlated subgraph pairs, which is a more general and harder problem.

VII. Conclusions

In this paper, we study the new problem of mining the set of frequent correlated subgraph pairs from a graph database, which finds potential usage in many important applications. We propose an efficient algorithm, FCP-Miner, to solve the problem. The algorithm employs a very effective skipping mechanism, which is devised based on a tight theoretical bound established on the minimum correlation between any two subgraphs in the answer set of a graph. We conduct extensive experiments, which verify that FCP-Miner is able to skip the processing of about 80% of the frequent subgraphs. Our algorithm is approximate, but it achieves a high quality of approximation, since the rate of missing results is very low (around 2-4% in most cases) and the missing pairs are shown to be marginally correlated. However, the efficiency gained from the approximation is that FCP-Miner is over an order of magnitude faster than the exact algorithm in most cases and it is also more scalable for various metrics.

Acknowledgment

The project was supported by the grant of RGC (No. 419008), Hong Kong SAR, China.

References

[1] "The Protein Data Bank," *http://www.rcsb.org/pdb/.*

[2] "Protein Databank in Europe," *http://www.ebi.ac.uk/pdbe/.*

[3] "PubChem," *http://pubchem.ncbi.nlm.nih.gov/.*

[4] "CADD Group Chemoinformatics Tools and User Services," *http://cactus.nci.nih.gov/.*

[5] "Drug Information Portal," *http://druginfo.nlm.nih.gov/.*

[6] "DrugBank," *http://www.drugbank.ca/.*

[7] "Trust Network Datasets - TrustLet," *http://www.trustlet.org/.*

[8] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *Proc. of PKDD*, 2000, pp. 13–23.

[9] M. Kuramochi and G. Karypis, "Frequent subgraph discovery." in *Proc. of ICDM*, 2001, pp. 313–320.

[10] C. Borgelt and M. R. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules," in *Proc. of ICDM*, 2002, p. 51.

[11] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Proc. of ICDM*, 2002, p. 721.

[12] J. Huan, W. Wang, and J. Prins, "Efficient mining of frequent subgraphs in the presence of isomorphism," in *Proc. of ICDM*, 2003, p. 549.

[13] X. Yan and J. Han, "Closegraph: mining closed frequent graph patterns," in *Proc. of KDD*, 2003, pp. 286–295.

[14] J. Huan, W. Wang, J. Prins, and J. Yang, "Spin: mining maximal frequent subgraphs from graph databases," in *Proc. of KDD*, 2004, pp. 581–586.

[15] L. T. Thomas, S. R. Valluri, and K. Karlapalem, "Margin: Maximal frequent subgraph mining," in *ICDM*, 2006, pp. 1097–1101.

[16] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: generalizing association rules to correlations," in *SIGMOD*, 1997, pp. 265–276.

[17] S. Morishita and J. Sese, "Traversing itemset lattice with statistical metric pruning," in *Proc. of PODS*, 2000, pp. 226–236.

[18] S. Ma and J. L. Hellerstein, "Mining mutually dependent patterns," in *ICDM*, 2001, pp. 409–416.

[19] H. Xiong, P.-N. Tan, and V. Kumar, "Hyperclique pattern discovery," *DMKD*, vol. 13, no. 2, pp. 219–242, 2006.

[20] H. Xiong, S. Shekhar, P.-N. Tan, and V. Kumar, "TAPER: A two-step approach for all-strong-pairs correlation query in large databases," *IEEE TKDE*, vol. 18, no. 4, pp. 493–508, 2006.

[21] H. Xiong, M. Brodie, and S. Ma, "Top-cop: Mining top-k strongly correlated pairs in large databases," in *ICDM*, 2006, pp. 1162–1166.

[22] Y. Ke, J. Cheng, and W. Ng, "Correlation search in graph databases," in *Proc. of KDD*. New York, NY, USA: ACM, 2007, pp. 390–399.

[23] Y. Ke, J. Cheng, and J. X. Yu, "Top-k correlative graph mining," in *Proc. of SDM*, 2009, pp. 1038–1049.

[24] H. Reynolds, *The analysis of cross-classifications*. New York: The Free Press, 1977.

[25] E. Lameijer, T. B$\ddot{a}$ck, J. Kok, and A. IJzerman, "Mining a chemical database for fragment cooccurrence: Discovery of "chemical clich$\acute{e}$s"," *Journal of Chemical Information and Modelling*, pp. 553–562, 2006.

[26] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. of STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, 1971, pp. 151–158.

[27] Y. Sakurai, S. Papadimitriou, and C. Faloutsos, "Braid: Stream mining through group lag correlations," in *SIGMOD Conference*, 2005, pp. 599–610.

[28] S. Papadimitriou, J. Sun, and P. S. Yu, "Local correlation tracking in time series," in *ICDM*, 2006, pp. 456–465.

[29] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proc. of KDD*, 2004, pp. 653–658.

[30] B. Bringmann, A. Zimmermann, L. D. Raedt, and S. Nijssen, "Don't be afraid of simpler patterns," in *Proc. of PKDD*, 2006, pp. 55–66.