On Mining Cross-Graph Quasi-Cliques[*]

Jian Pei
Simon Fraser University,
Canada
jpei@cs.sfu.ca

Daxin Jiang
State University of New York at
Buffalo, USA
djiang3@cse.buffalo.edu

Aidong Zhang
State University of New York at
Buffalo, USA
azhang@cse.buffalo.edu

ABSTRACT

Joint mining of multiple data sets can often discover interesting, novel, and reliable patterns which cannot be obtained solely from any single source. For example, in cross-market customer segmentation, a group of customers who behave similarly in multiple markets should be considered as a more coherent and more reliable cluster than clusters found in a single market. As another example, in bioinformatics, by joint mining of gene expression data and protein interaction data, we can find clusters of genes which show coherent expression patterns and also produce interacting proteins. Such clusters may be potential pathways.

In this paper, we investigate a novel data mining problem, *mining cross-graph quasi-cliques*, which is generalized from several interesting applications such as cross-market customer segmentation and joint mining of gene expression data and protein interaction data. We build a general model for mining cross-graph quasi-cliques, show why the complete set of cross-graph quasi-cliques cannot be found by previous data mining methods, and study the complexity of the problem. While the problem is difficult, we develop an efficient algorithm, *Crochet*, which exploits several interesting and effective techniques and heuristics to efficaciously mine cross-graph quasi-cliques. A systematic performance study is reported on both synthetic and real data sets. We demonstrate some interesting and meaningful cross-graph quasi-cliques in bioinformatics. The experimental results also show that algorithm *Crochet* is efficient and scalable.

Categories and Subject Descriptors: H.2.8 [Database Applications]: Data Mining

General Terms: Algorithms, Performance.

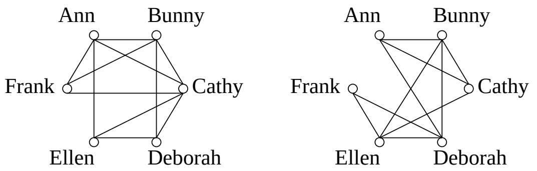
Keywords: Graph mining, mining multiple data sets, quasi-cliques, bioinformatics.

1. INTRODUCTION

In many applications, data is often collected, organized and stored in multiple sources. Many hidden patterns crossing multiple pieces of data cannot be found by mining only one single data set. Therefore, the advanced data analysis in practice calls for *joint mining of multiple data sets*, which can often discover interesting, novel, and reliable patterns that cannot be obtained solely from any single source.

EXAMPLE 1. (MOTIVATING EXAMPLE – CROSS-MARKET CUSTOMER SEGMENTATION). In marketing and customer relation management, customer segmentation is an important task, which partitions customers into groups according to their market behavior such as their purchase records and their responses to marketing campaigns. Customer segmentation in a single market has been extensively studied. However, it is interesting and informative to explore *cross-market customer segmentation*, which identifies groups of customers who have similar behavior in multiple markets. Customer groups found in cross-market customer segmentation can be more coherent and more reliable.

Consider the market behavior of six customers, Ann, Bunny, Cathy, Deborah, Ellen and Frank, in two markets, the financial product market and the consumer product market. In a specific market, the similarity among customers in market behavior can be modeled as a similarity graph. Each customer is a vertex in the graph, and two customers are connected by an edge if their behavior in the market is similar enough. Figure 1 shows the similarity graphs in the two markets.



G_f: similarity graph in financial product market

G_c: similarity graph in consumer product market

Figure 1: The similarity graph of six customers in two markets.

A close look at both similarity graphs may find that the subgraphs induced by the group of customers $S = \{$Ann, Bunny, Cathy, Deborah, Ellen$\}$ in both similarity graphs are interesting: they are quasi-cliques (i.e., the subgraphs are almost cliques). In each similarity graph, each customer in S is similar to at least 3 out of the 4 customers in the group. This observation may strongly suggest that S is a coherent and reliable customer group. ■

Example 1 motivates a novel problem of mining multiple graphs as follows. Consider multiple graphs about a set of objects which

[*] This research is supported in part by NSERC Grant 312194-05, NSF Grants IIS-0308001 and DBI-0234895, and NIH grant 1 P20 GM067650-01A1. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.
Copyright 2005 ACM 1-59593-135-X/05/0008 ...$5.00.

are the vertices of the graphs. We are interested in finding groups of objects such that the induced subgraph of each group of objects in each graph is almost a clique (thus called quasi-clique) – that is, in every graph, each object in a group is connected to at least a portion γ $(0 < \gamma \leq 1)$ of the other objects in the same group, where γ is a user-specified parameter. We call the problem *mining cross-graph quasi-cliques*.

Mining cross-graph quasi-cliques and its variations appear in many applications. We present below another interesting example in bioinformatics – joint mining gene expression data and protein interaction data.

EXAMPLE 2. (MOTIVATING EXAMPLE – JOINT MINING OF GENE EXPRESSION DATA AND PROTEIN INTERACTION DATA). From the gene expression data, we can find co-expressed genes, which are groups of genes that demonstrate coherent patterns on samples or against stimuli. On the other hand, from the protein interaction data, we can find groups of proteins which frequently interact with each other.

Genes and proteins are related – a protein is a product of a gene (however, one gene can produce more than one protein). If we can conduct a joint mining of both gene expression data and protein interaction data, then we may find the clusters of genes that are co-expressed and also their proteins interact.

Such cross-data set clusters found from the joint mining are interesting and meaningful in bioinformatics. First, both the gene expression data and the protein data are often very noisy. Clusters in a single source are often unreliable. The clusters confirmed by both data sets will strongly indicate the correlation/connection among the genes in a cluster, and thus are more reliable. Second, although highly related, gene expression data and protein interaction data still carry different biological meaning. The coincidence of co-expressed genes and interacting proteins is biologically significant. As indicated in [25], many pathways exhibit two properties: their genes exhibit a similar gene expression profile, and the protein products of the genes often interact.

Technically, the gene expression data and the protein interaction data can be modeled using a gene coherence graph G_g and a protein interaction graph G_p, respectively. In the gene coherence graph G_g, vertices are genes, and two genes are connected if their expression patterns are similar according to a user-specified similarity measure (e.g., Euclidean distance, Pearson's correlation coefficient, KL-distance [19], or pattern-based similarity measures [7, 29]). In the protein interaction graph G_p, the vertices are proteins, and two proteins are connected if they interact with each other. We use a surjective (i.e., onto) mapping $m(p) = g$ from proteins to genes to model the relationship between genes and proteins.

A group of proteins is interesting if each protein in the group interacts with most of the rest of the proteins in the same group, and in the set of genes producing the group of proteins, each gene is similar to most of the rest of the genes in the same group. They are *cross-graph quasi-cliques*. ■

As shown, mining multiple graphs may discover interesting patterns that cannot be found by conventional data mining approaches. Finding cliques in a graph is a problem that has been investigated for a long time. Computing quasi-cliques in one graph was also the topic of some recent studies, such as [1, 21]. One may wonder, "*Can the complete set of cross-graph quasi-cliques be mined easily by extending the existing algorithms for finding cliques or quasi-cliques in an 'integrated' graph?*"

EXAMPLE 3. (CAN CROSS-GRAPH QUASI-CLIQUES BE MINED FROM AN INTEGRATED GRAPH?). A natural thinking may be as follows. We can integrate the multiple graphs into one based on a similarity function between data objects. The integrated similar function combines the similarity between data objects in different data sets in some weighted manner. Then, we can find quasi-cliques in the integrated graph.

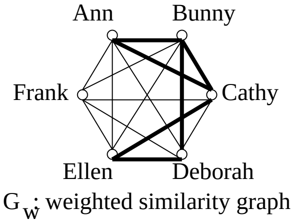


Figure 2: An integrated graph based on a weighted sum similarity function. The thick edges are of weight 2 and the thin edges are of weight 1.

Consider the two graphs in Example 1 again. Figure 2 shows an integrated graph based on a weighted sum similarity function. Any two objects are connected by an edge in the integrated graph if there is an edge between those two objects in either G_f or G_c in Figure 1. The weight of an edge is 2 if the edge appears in both G_f and G_c, otherwise, the weight is 1.

From the integrated graph, we cannot observe the group $S = \{$Ann, Bunny, Cathy, Deborah, Ellen$\}$ as a cluster. If we only consider all edges with weight 2, each of Ann, Bunny, Deborah and Ellen only connects to 2 of the other customers in S. If we consider all edges with weight 1 or above, then the 6 customers form a clique. Therefore, the integrated graph method does not work! ■

Although there are extensive studies on cliques and some recent studies on quasi-cliques, to the best of our knowledge, our study is the first one to address the following two issues at the same time.

- We investigate *mining from multiple graphs* the quasi-cliques. We propose a general model on the cross-graph quasi-cliques and the mining.
- We compute the *complete set of cross-graph* quasi-cliques. The previous studies focused on finding one quasi-clique (from one graph) with an optimization goal, such as maximizing the number of vertices in the clique. However, many data mining applications require the completeness of the answers.

Mining the complete set of quasi-cliques from multiple graphs is challenging. A naïve method may have to examine a huge number of possible combinations of vertices and edges over the graphs, which is computationally expensive or even prohibitive on large graphs (e.g., graphs with thousands of vertices and tens of thousands of edges).

Bearing the above challenges, in this paper, we tackle the problem of mining cross-graph quasi-cliques and make the following contributions.

- *We propose a novel and general model for the problem of mining cross-graph quasi-cliques*. We show that the cross-graph quasi-cliques are interesting and meaningful in applications.
- *We investigate the complexity of the problem and develop an efficient algorithm to tackle the problem*. We show that the problem is in #P-complete. We develop an efficient algorithm, *Crochet*, to mine cross-graph quasi-cliques. *Crochet* exploits several interesting and effective techniques and heuristics to prune the search space sharply.

- *We present a systematic performance study on* Crochet *to verify our design using both synthetic and real data sets*. The experimental results show that the cross-graph quasi-cliques are interesting in real applications and *Crochet* is efficient and scalable.

The remainder of the paper is organized as follows. In Section 2, we present the general model of mining cross-graph quasi-cliques, and also show the complexity of the problem. The algorithms are developed in Section 3. A systematic performance study is reported in Section 4. We discuss related work in Section 5. Section 6 concludes the paper.

2. MODEL AND PROBLEM DEFINITION

In this section, we propose a general model for cross-graph quasi-clique mining and show the complexity of the problem.

2.1 Quasi-Complete Graph

In this paper, we consider *simple graphs* only, i.e., the graphs without self-loops or multi-edges. For graph G, $V(G)$ and $E(G)$ denote the sets of vertices and edges of G, respectively.

For vertices $u, v \in V(G)$, let $d(u, v)$ be the number of edges in the shortest path between u and v. Trivially, $d(u, u) = 0$. A graph is called *connected* if $d(u, v) < \infty$ for any $u, v \in V(G)$.

For a vertex $u \in V(G)$, $N(u)$ is the set of neighbors of u, i.e., $N(u) = \{v|(u, v) \in E(G)\}$. Moreover, we define $N^k(u) = \{v|d(u, v) \leq k\}$ for $(k \geq 1)$. Clearly, $N^{(|V(G)|-1)}(u)$ is the set of vertices that are connected to u. We also denote the set by $N^*(u)$. In a connected graph, $N^*(u) = V(G)$.

In graph G, let $U \subseteq V(G)$ be a subset of vertices. The *subgraph induced on* U, denoted by $G(U)$, is the subgraph of G whose vertex-set is U and whose edge-set consists of all edges in G that have both endpoints in U, i.e., $G(U) = (U, E_U)$, where $E_U = \{(u, v)|(u, v) \in E(G) \wedge u, v \in U\}$.

A *complete graph* is a graph such that every pair of vertices is joined by an edge. In a graph G, a subset of vertices $S \subseteq V(G)$ is a *clique* if the subgraph induced on S, i.e., $G(S)$, is a complete graph, and no proper superset of S has this property. Please note that, there can be more than one clique in a graph, and the cliques may not be exclusive. That is, two cliques may share some common vertices.

DEFINITION 2.1. (QUASI-COMPLETE GRAPH AND QUASI-CLIQUE). A connected graph G is a γ*-complete graph* $(0 < \gamma \leq 1)$ if every vertex in the graph has a degree at least $\gamma \cdot (|V(G)| - 1)$.

In a graph G, a subset of vertices $S \subseteq V(G)$ is a γ*-quasi-clique* $(0 < \gamma \leq 1)$ if $G(S)$ is a γ-quasi-complete graph, and no proper superset of S has this property. ■

Clearly, a 1-quasi-complete graph is a complete graph, and a 1-quasi-clique is a clique.

The degree of a vertex must be an integer. A γ-quasi-complete graph G must be connected. That is, necessarily, $\gamma \cdot (|V(G)| - 1) \geq 1$. Hence, we have $\gamma \geq \frac{1}{|V(G)|-1}$. Moreover, we have the following result.

PROPOSITION 1. (ANTI-MONOTONICITY OF COMPLETE GRAPHS). *In a graph G, let $S \subseteq V(G)$. If $G(S)$ is a complete graph, then, for any subset $S' \subset S$, $G(S)$ is also a complete graph.* ■

In general, the anti-monotonicity does not hold for quasi-complete graphs. That is, for a γ-quasi-complete graph G $(0 < \gamma < 1)$, $G(S)$ may not be a γ-quasi-complete graph for an $S \subset V(G)$.

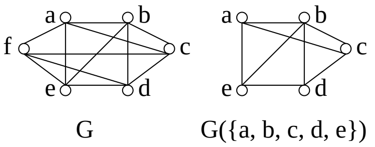


Figure 3: A graph and an induced subgraph.

EXAMPLE 4 (QUASI-COMPLETE GRAPH). Consider graph G in Figure 3. It is a 0.8-quasi-complete graph, since every vertex has a degree of $4 = (6 - 1) \times 0.8$.

Interestingly, a subgraph induced on any subset of 5 vertices is not a 0.8-quasi-complete graph. As an example, the subgraph induced on $\{a, b, c, d, e\}$ is also shown in Figure 3. Vertices a, c, d and e in the induced subgraph have a degree of $3 < (5 - 1) \times 0.8 = 3.2$. ■

In γ-quasi-complete graphs, parameter γ controls the compactness of the graph. That is, with a larger γ, each vertex joins to more other vertices and thus the graph is more compact. One measure of the compactness of a graph is the diameter. The *diameter of* G, denoted by $diam(G)$, is defined as $diam(G) = \max_{u,v \in V(G)}\{d(u, v)\}$. It is interesting to examine the relationship between the diameter of a γ-quasi-complete graph and γ.

THEOREM 1 (DIAMETER OF QUASI-COMPLETE GRAPH). *Let G be a γ-quasi-complete graph such that $n = |V(G)| > 1$.*

$$diam(G) \begin{cases} = 1 & \textit{if } 1 \geq \gamma > \frac{n-2}{n-1} \\ \leq 2 & \textit{if } \frac{n-2}{n-1} \geq \gamma \geq \frac{1}{2} \\ \leq 3\lfloor \frac{n}{\gamma(n-1)+1} \rfloor - 3 & \textit{if } \frac{1}{2} > \gamma \geq \frac{2}{n-1} \textit{ and} \\ & n \bmod (\gamma(n-1)+1) = 0 \\ \leq 3\lfloor \frac{n}{\gamma(n-1)+1} \rfloor - 2 & \textit{if } \frac{1}{2} > \gamma \geq \frac{2}{n-1} \textit{ and} \\ & n \bmod (\gamma(n-1)+1) = 1 \\ \leq 3\lfloor \frac{n}{\gamma(n-1)+1} \rfloor - 1 & \textit{if } \frac{1}{2} > \gamma \geq \frac{2}{n-1} \textit{ and} \\ & n \bmod (\gamma(n-1)+1) \geq 2 \\ \leq n-1 & \textit{if } \gamma = \frac{1}{n-1} \end{cases}$$

The upper bounds are realizable.

Proof sketch. When $1 \geq \gamma > \frac{n-2}{n-1}$, the degree of each vertex is greater than $(n - 2)$, and hence must be $(n - 1)$. Thus, G is a complete graph and $diam(G) = 1$.

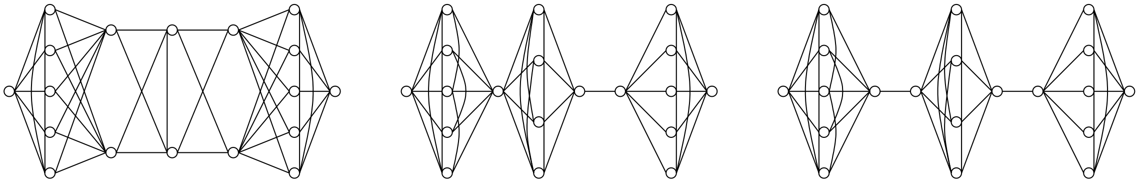
When $\frac{n-1}{n-2} \geq \gamma \geq \frac{1}{2}$, for any vertices $u, v \in V(G)$, $|\{u, v\} \cup N(u) \cup N(v)| \geq n$. Thus, $diam(G) = 2$.

When $\frac{1}{2} > \gamma \geq \frac{2}{n-1}$, the situations are complicated. Consider vertices $u, v \in V(G)$ such that the shortest path between u and v has length $l = diam(G)$. $V(G)$ can be partitioned into $(l + 1)$ exclusive groups $S_1, \ldots, S_{l+1}$: a vertex $w \in S_i$ $(1 < i \leq (l + 1))$ if and only if $d(u, w) = (i - 1)$.

Clearly, $\cup_{1 \leq i \leq (l+1)} S_i = V(G)$, otherwise, $diam(G) > l$. A critical fact is that, for any vertex $w \in S_i$ $(1 \leq i \leq (i + 1))$, $N(w) \subseteq S_{i-1} \cup S_i \cup S_{i+1}$. That means $|S_{i-1} \cup S_i \cup S_{i+1}| = |S_{i-1}| + |S_i| + |S_{i+1}| \geq deg(w) + 1 \geq \gamma(n-1) + 1$. Moreover, we have $|S_1| = 1$, $|S_2| = \gamma(n - 1)$, and $|S_l| + |S_{l+1}| \geq \gamma(n-1) + 1$ (otherwise, there exists at least one vertex v such that $deg(v) < \gamma(n - 1)$). The inequations in the theorem can be proved using the above inequations. Limited by space, we omit the details here.

When $\gamma = \frac{1}{n-1}$, some vertices have degree 1. In the worst case, the graph can be a path, and thus $diam(G) = (n - 1)$. Recall that a quasi-complete graph must be connected. $\gamma \geq \frac{1}{n-1}$.

The bounds can be shown realizable. For example, Figure 4 shows three cases that the bounds are realized for $\frac{1}{2} > \gamma \geq \frac{2}{n-1}$. ■



n=18, r=5/17, d=6 n=19, r=5/19, d=7 n=20, r=1/4, d=8

Figure 4: The cases where $\frac{1}{2} > \gamma \geq \frac{2}{n-1}$ and the upper bounds in Theorem 1 are realized.

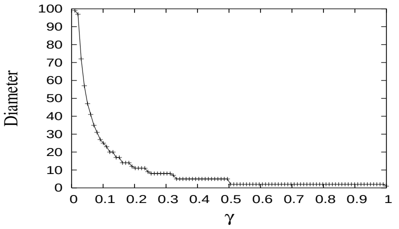


Figure 5: The change of diameter on γ ($|V(G)| = 100$).

The theorem is important, which discloses the effect of γ. Figure 5 shows the trend of diameter on γ, where the number of vertices is set to 100. We can observe the following. First, when γ is reasonably large, i.e., 0.5 or up, the γ-quasi-complete graph is compact, i.e., the diameter is very small, no more than 2. Second, when γ is small, the upper bound of the diameter of the quasi-complete graph is approximately in portion to $\frac{1}{\gamma}$, which is intuitive. Third, when γ is small, the quasi-complete graph can be a series of small clusters. When $\gamma = \frac{1}{n-1}$, in the worst case, the quasi-complete graph can be a path of n vertices. Based on the above analysis, a user may be often interested in quasi-complete graphs with a reasonably large γ value, such as $\gamma \approx 0.5$ or larger. In such cases, the diameter is bounded by a small integer.

2.2 Cross-Graph Quasi-Clique

Intuitively, a cross-graph quasi-clique is a maximal set of vertices whose induced graphs in all the graphs are quasi-complete subgraphs.

DEFINITION 2.2 (CROSS-GRAPH QUASI-CLIQUE). Let U be a set of vertices and $G_1, \ldots, G_n$ be n graphs such that $V(G_i) = U$ $(1 \leq i \leq n)$. For parameters $\gamma_1, \ldots, \gamma_n$ $(0 < \gamma_i \leq 1)$, a subset $S \subseteq U$ of vertices is called a *cross-graph quasi-clique* (*CGQC* for short) if $G_i(S)$ is a γ_i-quasi-complete graph for all $(1 \leq i \leq n)$ and there is no proper superset of S has the property. ■

Clearly, in Definition 2.2, when $n = 1$, a cross-graph quasi-clique on a single graph is simply a quasi-clique in the graph. Therefore, the concept of cross-graph quasi-clique is a generalization of quasi-clique crossing multiple graphs. However, when $n > 1$, a cross-graph quasi-clique may not be a quasi-clique in any graph due to the maximality requirement.

EXAMPLE 5 (MAXIMALITY). Consider graphs G_1 and G_2 in Figure 6. Suppose $\gamma_1 = \gamma_2 = 0.5$. Then, $S = \{a, b, d\}$ is a cross-graph quasi-clique. However, S is not a 0.5-quasi-clique in either G_1 or G_2. In G_1, S is a proper subset of $\{a, b, d, e\}$ which is a 0.5-quasi-clique. In G_2, S is also a proper subset of $\{a, b, c, d\}$ which is a 0.5-quasi-clique. ■

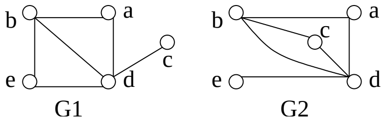


Figure 6: A cross-graph quasi-clique may not be quasi-cliques in any individual graphs due to the maximality requirement.

A cross-graph quasi-clique can be insignificant in data analysis if it contains a very small number of vertices. For example, a single vertex itself is a (trivial) quasi-complete graph for any γ. To avoid such a triviality, a user may specify a minimum number of vertices in the cross-graph quasi-cliques. Only cross-graph quasi-cliques large enough should be returned.

Problem Definition (Mining Cross-Graph Quasi-Cliques). For a given set of graphs $G_1, \ldots, G_n$ on a set of vertices U (i.e., $V(G_1) = \cdots = V(G_n) = U$), parameters $\gamma_1, \ldots, \gamma_n$ $(0 < \gamma_i \leq 1)$, and a minimum size threshold min_s, the problem of *mining cross-graph quasi-cliques* is to find the complete set of cross-graph quasi-cliques that each has at least min_s vertices. ■

In some cases, such as joint mining of gene expression data and protein interaction data (Example 2), the sets of vertices in different graphs are different, and there exist mappings between sets of vertices in different graphs. Our basic model of mining cross-graph quasi-cliques can be extended to handle such cases.

DEFINITION 2.3. (CROSS-GRAPH QUASI-CLIQUE WITH MAPPING). Let $G_1, \ldots, G_n$ be n graphs and $f_2, \ldots, f_n$ be n functions such that f_i $(2 \leq i \leq n)$ is from $V(G_1)$ to $V(G_i)$. For parameters $\gamma_1, \ldots, \gamma_n$ $(0 < \gamma_i \leq 1)$, a subset $S \subseteq V(G_1)$ of vertices in G_1 is called a *cross-graph quasi-clique with mapping* (*CGQC(M)* for short) if (1) $G_1(S)$ is a γ_1-quasi-complete graph; (2) for $S_i = \{f_i(v) | v \in S\}$, $G_i(S_i)$ is a γ_i-quasi-complete graph; and (3) there is no proper superset of S has properties (1) and (2). ■

Problem Definition (Mining Cross-Graph Quasi-Cliques with Mapping). For a given set of graphs $G_1, \ldots, G_n$ and functions f_2, $\ldots$, f_n such that $f_i : V(G_1) \rightarrow V(G_i)$ $(2 \leq i \leq n)$, parameters $\gamma_1, \ldots, \gamma_n$ $(0 < \gamma_i \leq 1)$, and a minimum size threshold min_s, the problem of *mining cross-graph quasi-cliques with mapping* is to find the complete set of CGQC(M)'s that each has at least min_s vertices. ■

THEOREM 2 (COMPLEXITY). *The problem of counting the number of cross-graph quasi-cliques is in #P-Complete.*

Proof sketch. We prove by restriction. That is, we show that the problem of counting the number of cross-graph quasi-cliques contains a #P-Complete problem as a special case. In fact, the problem of counting the number of cliques from one graph is in #P-Complete [14], and is a special case of the problem of counting the number of cross-graph quasi-cliques where $n = 1$ and $\gamma = 1$. ■

Thus, the problem of mining (i.e., enumerating) the complete set of cross-graph quasi-cliques is NP-hard in the worst case.

```
Input: graphs $G_1, \ldots, G_n$; $\gamma_1, \ldots, \gamma_n$; mapping
    functions $f_2, \ldots, f_n$; minimum size threshold $min_s$;
Output: the complete set of cross-graph quasi-cliques;
Method:
    // Phase 1: mining $G_1$
    in $G_1$, compute the complete set of $\gamma_1$-quasi-complete
    subgraphs $H$ that has at least $min_s$ vertices;
    // Phase 2: jointly mining
    for each $\gamma_1$-quasi-complete subgraph $H$, in the
    $|V(H)|$ descending order
      let cross-graph-quasi-clique = true;
      if $H$ has a proper superset as a cross-graph
      quasi-clique then continue;
      for $i = 2$ to $n$
        if $G_i(V(H))$ is not a $\gamma_i$-quasi-complete graph
        then cross-graph-quasi-clique = false, break;
      end-for
      if (cross-graph-quasi-clique == true) then
      output $V(H)$ as a cross-graph quasi-clique;
    end-for
```

Figure 7: The rudimentary algorithm.

3. ALGORITHMS

In this section, we develop algorithms for mining cross-graph quasi-cliques. The algorithms target at the applications where there are a small number of large graphs. We assume that the graphs can be held into main memory.

We present two algorithms. The first algorithm is rudimentary. The second algorithm, *Crochet*, exploits several smart and effective techniques to achieve efficient mining.

3.1 A Rudimentary Algorithm

As illustrated in Examples 4 and 5, a cross-graph quasi-clique may not be a quasi-clique in any graph, and an induced subgraph on a subset of a quasi-clique may not even be a quasi-complete graph. Hence, we cannot find the quasi-cliques in each graph and take the intersection. Instead, *we have to take a joint mining approach.*

By definition, a cross-graph quasi-clique must be a quasi-complete graph in every graph by the mapping. Thus, a rudimentary algorithm works in two steps: *mining* G_1 and *jointly mining*, as shown in Figure 7.

In the first step, we mine the complete set of γ_1-quasi-complete graphs in G_1 which have at least min_s vertices. In other words, in graph G_1, we find the subsets of vertices S such that the subgraph induced on S is a γ_1-quasi-complete graph.

In the step of joint mining, for each subset of vertices S found in the first step, we check whether $G_i(S)$ is still a γ_i-quasi-complete graph for all $(2 \leq i \leq n)$. Only maximal subsets passing the tests are output as the cross-graph quasi-cliques.

Please note that the rudimentary algorithm never searches the complete set of quasi-complete subgraphs in all the graphs. Instead, to prune the search space, it exploits the fact that a cross-graph quasi-clique must form a quasi-complete subgraph in G_1 and must be a maximal subset having the properties in Definition 2.2.

The rudimentary algorithm may not be efficient in mining large graphs due to two reasons. First, *it still has to compute the complete set of quasi-complete subgraphs in G_1, since it conducts the joint mining as the second step.* If G_1 is large and dense (i.e., it has many edges), then there can be many such quasi-complete subgraphs. Computing all of them can be expensive. *Can we compute as less quasi-complete subgraphs in G_1 as possible?*

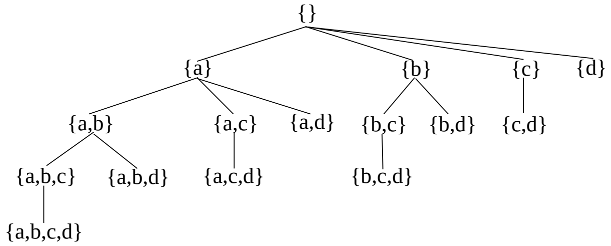


Figure 8: A set enumeration tree.

Second, *the rudimentary algorithm mines the whole graphs.* If the graphs are large, it can be costly. A careful check can find that, some vertices and edges in the graphs, such as the vertices with low degrees, cannot be a part of a cross-graph quasi-clique or the induced subgraphs. Such parts should be pruned as early as possible so that the graphs can be reduced. Mining smaller graphs can definitely improve the efficiency. In order to reduce the graphs, we have to consider multiple graphs at the very beginning. *Can we reduce the graphs aggressively to speed up the mining?*

In summary, the major drawback of the rudimentary algorithm is that *it conducts the joint mining late*. We need to exploit "aggressive" joint mining of multiple graphs.

3.2 Algorithm *Crochet*

To make our presentation clear and easy to follow, we first examine a basic case of cross-graph quasi-clique mining – mining from two graphs G_1 and G_2 such that $V(G_1) = V(G_2) = U$. Then, we discuss how to extend the basic case to mine multiple graphs and handle mapping functions other than bijections.

3.2.1 General Idea and Framework

In order to efficiently mine the complete set of cross-graph quasi-cliques, we have to address two issues as follows. First, the *correctness and completeness.* That is, how can we find a systematic way to find *all* the cross-graph quasi-cliques *without duplicates*? Second, the *efficiency.* That is, how can we find rules to prune futile search subspaces (i.e., the subspaces where there is no cross-graph quasi-clique at all) and heuristics to speed up the mining?

To address the issue of correctness and completeness, we compute the complete set of cross-graph quasi-cliques by enumerating the subsets of vertices systematically and pruning the unfruitful subsets.

Given a set S of n elements and a total order $\prec$ on S, the complete set of various subsets of S (i.e., 2^S) can be enumerated systematically using a *set enumeration tree* [24]. For example, the set enumeration tree for set $\{a, b, c, d\}$ with respect to order $a \prec b \prec c \prec d$ is shown in Figure 8.

In a set enumeration tree of vertices, each node is a subset of vertices. Some nodes are cross-graph quasi-cliques. Algorithm *Crochet* conducts a depth-first search on the set enumeration tree of vertices to find the cross-graph quasi-cliques, which can identify the complete set of answers.

To address the concern on efficiency, at each step of the depth-first search, *Crochet* employs the following techniques.

- *Aggressively reducing graphs. Crochet* removes edges and vertices, and even combines graphs as long as the correct mining results are retained. By reducing graphs, *Crochet* can work on smaller graphs and thus can mine cross-graph quasi-cliques faster.
- *Carefully choosing the order of search.* In the depth-first search of a set enumeration tree, a node in the tree may have

multiple children. The order that we use to search the children may have a substantial effect on the efficiency. *Crochet* uses an effective heuristic to dynamically determine the order of children based on the information from multiple graphs.

- *Sharply pruning futile subtrees.* If a graph has many vertices, the set enumeration tree can be huge. *Crochet* actively detects whether a subtree has the potential to have some cross-graph cross-cliques before it really searches the nodes in the subtree. If a subtree is futile, i.e., it is impossible to contain any node of cross-graph quasi-cliques, then the subtree should be pruned as early as possible.

3.2.2 Reducing Graphs

Reducing Vertices

Some vertices in the graphs can be removed if their degrees are too small or they are not connected to enough other vertices to form a quasi-complete graph.

LEMMA 3.1 (REDUCING VERTICES). *For vertex u in graph G_i ($i = 1$ or 2), if $deg(u) < \gamma_i \cdot (min_s - 1)$ or $|N^k(u)| < (min_s - 1)$, where k is the upper bound of $diam(G)$ in Theorem 1, then u and the edges having u as an endpoint can be removed from both G_1 and G_2 and all cross-graph quasi-cliques in G_1 and G_2 are retained.*

Proof. If $deg(u) < \gamma_i \cdot (min_s - 1)$, u fails the requirement on the minimum degree in a quasi-complete graph in G_i. If $|N^k(u)| < (min_s - 1)$, u cannot be a vertex in any subgraph which has a diameter k and also has at least min_s vertices. Hence, in either of the two cases, u cannot be a vertex in any quasi-complete subgraph in G_i. That means u cannot be a vertex in any cross-graph quasi-clique. Thus, u as well as the edges having u as an endpoint can be removed from G_1 and G_2, and the mining results will not be affected. ■

Lemma 3.1 can be applied iteratively to reduce the graphs, until no vertex or edge can be removed. The checking of degrees is simple. However, dynamically maintaining $N^k(u)$ for every vertex u can be costly if k is large. Fortunately, as shown in Theorem 1, the upper bound of the diameter of a γ-quasi-complete graph is pretty small if γ is not too small. Moreover, as discussed before, a user is often interested in only the cross-graph quasi-cliques with respect to reasonably large γ values, e.g., $\gamma \approx 0.5$ or larger. In our experiments, when $\gamma \geq 0.5$, Lemma 3.1 often improves the efficiency by a factor of at least 20%.

Reducing Edges

Between G_1 and G_2, if there is a $\gamma_i = 1$ ($i = 1$ or 2), then we can remove some edges in the other graph according to the edges in G_i.

LEMMA 3.2 (REDUCING EDGES). *When $\gamma_i = 1$, an edge $(u, v) \in E(G_{3-i})$ can be removed if $(u, v) \notin E(G_1)$, and all cross-graph quasi-cliques from G_1 and G_2 are retained.*

Proof. Suppose $\gamma_1 = 1$. If $(u, v) \notin E(G_1)$, u and v cannot be in a complete subgraph in G_1, and thus cannot be in a cross-graph quasi-clique. Removing (u, v) from $E(G_2)$ will not affect any cross-graph quasi-cliques. ■

We can apply Lemma 3.2 to reduce the graphs by scanning the edges of the two graphs once, if their edges are sorted consistently.

Combining Graphs

In the case $\gamma_1 = \gamma_2 = 1$, we can combine the two graphs G_1 and G_2 into one graph G as follows. The vertices in G is the same set of vertices in G_1 and G_2, i.e., $V(G) = V(G_1) = V(G_2)$; (u, v) is an edge in G if and only if (u, v) is an edge in both G_1 and G_2, i.e., $E(G) = E(G_1) \cap E(G_2)$. Then, the problem of mining cross-graph quasi-cliques from the two graphs can be reduced to mining cliques in the combined graph.

LEMMA 3.3 (COMBINING GRAPHS). *When $\gamma_1 = \gamma_2 = 1$, let G be a combined graph such that $V(G) = V(G_1) = V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$. A set of vertices S is a cross-graph quasi-clique in G_1 and G_2 if and only if S is a clique in G.*

Proof. Intuitively, when $\gamma_1 = \gamma_2 = 1$, S is a cross-graph quasi-clique means that the induced subgraph on S in G_1 and G_2 are both complete sub-graphs. In other words, every edge in the induced subgraph on S in one graph (e.g., G_1) must also appear in the other graph (e.g., G_2). ■

3.2.3 Searching and Pruning

As discussed in Section 3.2.1, *Crochet* conducts a depth-first search on a set enumeration tree of vertices. If a graph has many vertices, the set enumeration tree can be huge. Therefore, one critical part of *Crochet* is to prune the futile subtrees as early as possible.

Basically, three issues have to be addressed.

- At a node X in the set enumeration tree, what are the vertices $u \in (V(G_1) - X)$ that should be used to extend X to its children and may likely lead to a cross-graph quasi-clique?
- In what situation is a subtree rooted at the current node in the set enumeration tree futile (i.e., there is no cross-graph quasi-clique in the subtree) and thus can be pruned?
- A node in the set enumeration tree may have multiple children. In which order should the children be searched so as the efficiency is likely high?

We answer the above three questions one by one.

Generating Children

Let us consider a node $X \subseteq V(G_1)$ in the set enumeration tree such that $X \neq \emptyset$. The following lemma indicates the set of vertices which can be used to generate the children of X that may lead to some cross-graph quasi-cliques.

LEMMA 3.4 (CANDIDATE VERTICES). *Let $X \neq \emptyset$ be a subset of vertices. If $C \supset X$ is a cross-graph quasi-clique, then for every vertex $u \in (C - X)$,*

$$u \in \bigcap_{v \in X, i=1,2} N_{G_i}^{k_i}(v),$$

where k_i is the upper bound of diameter of complete γ_i-quasi-complete graph in G_i given by Theorem 1.

Proof sketch. Following Theorem 1, for any $u \in (C - X)$ and $v \in X$, $d(u, v)$ in G_i is bounded by the upper bound of the diameter. Thus, we have the lemma. ■

For a node X in the set enumeration tree, Lemma 3.4 gives the initial set of *candidate vertices*. Moreover, as required by the set enumeration tree, only the vertices behind the last vertex in X in the order of vertices should be taken.

Pruning Futile Children and Subtrees

The initial set of candidate vertices can be reduced further. The central idea is as follows. Let Y be the initial set of candidate vertices given by Lemma 3.4. If there is no cross-graph quasi-clique in $G_1(X \cup Y)$ and $G_2(X \cup Y)$, then X cannot be in any cross-graph quasi-cliques. In other words, the vertices in Y should not be used to generate children of X since they are futile.

LEMMA 3.5 (PROJECTION). *Let $X \subset V(G_1)$ be a node in the set enumeration tree and Y be the initial set of candidate vertices given by Lemma 3.4. For any C such that $X \subset C \subseteq (X \cup Y)$, C is a cross-graph quasi-clique in G_1 and G_2 if and only if C is a cross-graph quasi-clique in $G_1(X \cup Y)$ and $G_2(X \cup Y)$.*

Proof sketch. The necessity is straightforward. To show the sufficiency, suppose C is a cross-graph quasi-clique in $G_1(X \cup Y)$ and $G_2(X \cup Y)$, but not in G_1 and G_2. Thus, there must be a $C' \supset C$ such that C' is a cross-graph quasi-clique in G_1 and G_2. However, according to the construction of X and Y, $X \subset C' \subseteq (X \cup Y)$. Thus, C' must be a cross-graph quasi-clique in $G_1(X \cup Y)$ and $G_2(X \cup Y)$, which leads to a contradiction. ■

$G_i(X \cup Y)$ is called the *projection* of G_i on X. Based on Lemma 3.5, we can recursively apply the vertex reduction (Lemma 3.1) to prune the projections. We denote the set of vertices in the projections after the pruning by $P(X)$. Clearly, in the set enumeration tree, only the children of X in the form of $X \cup \{u\}$ $u \in (P(X) - X)$ should be considered.

Moreover, in some situations, the whole subtree rooted at X cannot contain any cross-graph quasi-cliques. The following lemma lists four cases.

LEMMA 3.6 (PRUNING RULES). *Let X be a node in the set enumeration tree. The subtree rooted at X does not contain any cross-graph quasi-clique if (1) $|P(X)| < min_s$; (2) $X \not\subseteq P(X)$; (3) $P(X) \subset C$ where C is a cross-graph quasi-clique already found; or (4) X is not a clique in G_i if $\gamma_i = 1$.*

Proof sketch. The first pruning rule follows the requirement on the minimum number of vertices in cross-graph quasi-cliques. That is, if the sets of vertices in the subtree are too small, there is no hope to find a significant cross-graph quasi-clique and thus the subtree can be pruned.

The second rule follows the construction of the set enumeration tree. That is, some vertex in X can be pruned by edge reduction and vertex reduction. Then, there is no hope to get a cross-graph quasi-clique from the subtrees which are superset of X.

The third rule follows the requirement of maximality for cross-graph quasi-cliques. In other words, the cross-graph quasi-cliques containing X as a subset should be found in different branches of the set enumeration tree instead of the subtree rooted at X.

The last rule follows Proposition 1. Since $\gamma_i = 1$ and X is not a clique in G_i, X cannot be in any cross-graph quasi-clique. ■

If one of the four conditions specified in Lemma 3.6 happens, the subtree rooted at X should be pruned.

Ordering Children and Identifying Cross-Graph Quasi-Cliques

Intuitively, a vertex with a high degree is likely a member of a cross-graph quasi-clique. Heuristically, we can use $\theta(v) = \frac{ldeg(v)}{\gamma}$ to measure how well a vertex satisfies the γ-quasi-complete graph requirement, where $ldeg(v)$ is the degree of v in the induced graph on the current node X in the set enumeration tree. The vertices can be sorted in the $\theta(v)$ descending order. However, a vertex may have different θ value in different graphs. An observation is that how well a vertex is connected to the others crossing the graphs is bounded by the minimum $\theta(v)$ value in the graphs. Therefore, we have the following heuristic.

HEURISTIC 1 (ORDERING VERTICES). *The children vertices of a node in the set enumeration tree can be explored in the $\theta_{\min}(v)$ descending order, where $\theta_{\min}(v) = \min_{G_1,G_2}\{\theta(v)\}$.* ■

The experimental results in Section 4 show that the heuristic accomplishes good performance in practice. However, since it is a heuristic, there is no theoretical guarantee that the rule always achieves the optimal efficiency.

```
Input: graphs $G_1$, $G_2$; $\gamma_1$, $\gamma_2$; minimum size
    threshold $min_s$;
Output: the complete set of cross-graph quasi-cliques;
Method:
    // graph reduction
    apply edge reduction (Lemma 3.2), vertex
    reduction (lemma 3.1), and combine graphs (Lemma 3.3)
    if possible;
    if graphs can be combined then compute the complete set
    of cliques in the combined graph; exit;
    let $G_1$ and $G_2$ denote the reduced graphs;
    // depth-first search
    for each vertex $v \in V(G_1)$ in $\theta_{\min}(v)$ descending
    order // Heuristic 1
      let $X = \{x\}$
      call recursive-mine$(X, G_1, G_2)$;
    end for

Function recursive-mine$(X, G_1, G_2)$
    compute $P(X)$ according to Lemma 3.4;
    // graph reduction
    let $G_i = G_i(P(X))$; // Lemma 3.5
    apply vertex reduction (lemma 3.1);
    let $G_1$ and $G_2$ denote the reduced graphs;
    if at least one condition in Lemma 3.6 holds
    then return(0);
    // depth-first search
    let unsubsumed$= 1$;
    for each vertex $v \in P(X) - X$, in $\theta_{\min}(v)$ descending
    order // Heuristic 1
      call recursive-mine$(X \cup \{v\}, G_1, G_2)$;
      if the returned value is 1 then unsubsumed$= 0$;
    end for
    if unsubsumed is 0 then return$(1)$;
    else // Lemma 3.7
      if $G_i(X)$ is a $\gamma_i$-quasi-complete graph then
        output $X$ as a cross-graph quasi-clique and return$(1)$;
      else return$(0)$;
```

Figure 9: Algorithm *Crochet*: the basic case.

After searching the subtree rooted at X, we can determine that X is a cross-graph quasi-clique if $G_1(X)$ and $G_2(X)$ are both quasi-complete graphs and there is no cross-graph quasi-clique in the subtree of X.

LEMMA 3.7 (DETERMINATION OF CGQC). *Let X be a node in the set enumeration tree. X is a cross-graph quasi-clique if and only if $G_1(X)$ and $G_2(X)$ are both quasi-complete graphs and there exists no cross-graph quasi-clique C such that $X \subset C \subseteq P(X)$.* ■

Algorithm *Crochet* is summarized in Figure 9.

3.2.4 Mining More Than Two Graphs

The basic *Crochet* algorithm can be straightforwardly extended to handle more than two graphs. If there are more than one graph with $\gamma = 1$, we can combine them into one graph according to Lemma 3.3. The benefit of combining all the graphs having $\gamma = 1$ into one is twofold. First, it reduces the number of graphs. Second, it can be shown that, for the combined graph G, $|E(G)| \leq$

$\min_{1 \leq i \leq l}\{|E(G_i)|\}$. In words, the combined graph reduces the edges and thus may have a better chance to use the edge-reduction (Lemma 3.2) to further reduce edges in other graphs.

3.2.5 Handling Non-Bijective Functions

Now, let us consider the problem of mining cross-graph quasi-cliques with mapping, where the mapping from $V(G_1)$ to $V(G_i)$ $(i > 1)$ may not be bijective. *Crochet* can handle the non-bijective functions with minor extensions.

Consider mining cross-graph quasi-cliques from graphs G_1 and G_2 such that $V(G_1) \neq V(G_2)$. Let $f_2 : V(G_1) \rightarrow V(G_2)$ be the mapping. For any vertex $v \in V(G_2)$, the pre-images of v under f_2 is denoted as $f_2^{-1}(v) = \{u|(u \in V(G_1)) \wedge (f_2(u) = v)\}$.

Clearly, a vertex v cannot be in any cross-graph quasi-clique if $f_2(v)$ does not appear in G_2. Moreover, any vertex in G_2 is irrelevant to any cross-graph quasi-clique if there exists no vertex u in G_1 such that $f_2(u) = v$. Those vertices can be removed and the mining results will not be affected.

LEMMA 3.8. (REMOVING VERTEX WITH NO IMAGE OR NO PRE-IMAGE). *A vertex $u \in V(G_1)$ and the edges in G_1 having u as an endpoint can be removed if $f_2(u) \notin V(G_2)$. A vertex $v \in V(G_2)$ and the edges in G_2 having v as an endpoint can be removed from G_2 if $f_2^{-1}(v) = \emptyset$. The removal of those vertices and edges will not affect the mining of cross-graph quasi-cliques.* ■

For non-bijective mapping, all the above discussion about *Crochet* still holds by replacing $u \in V(G_i)$ by $f_i(u)$, except for edge reduction (Lemma 3.2) and combining graphs (Lemma 3.3).

For edge reduction (Lemma 3.2), there are two cases. First, suppose $\gamma_1 = 1$. For any edge $(u, v) \in E(G_i)$ $(i > 1)$, if there exists no edge $(u', v') \in E(G_1)$ such that $u' \in f_i^{-1}(u)$ and $v' \in f_i^{-1}(v)$, then (u, v) can be safely removed from G_i and all cross-graph quasi-cliques are retained, since such an edge cannot contribute to the construction of any cross-graph quasi-clique.

Second, suppose $\gamma_{i_0} = 1$. For any edge $(u, v) \in E(G_1)$, if $(f_{i_0}(u), f_{i_0}(v)) \notin E(G_{i_0})$, then (u, v) can be removed from graph G_1 and all cross-graph quasi-cliques are retained. Moreover, for any edge $(u, v) \in E(G_i)$ $(i > 1, i \neq i_0)$, if there exists no edge $(u', v') \in E(G_{i_0})$ such that $u' \in f_{i_0}(f_i^{-1}(u))$ and $v' \in f_{i_0}(f_i^{-1}(v))$, then (u, v) can be removed from graph G_i and the cross-graph quasi-cliques are retained. The rationale is similar. Limited by space, we omit the formal proof here.

For combining graphs (Lemma 3.3), in order to handle non-bijective mapping, we revise the definition of combined graph as follows. Suppose graphs $G_{i_1}, \ldots, G_{i_l}$ are with $\gamma_{i_j} = 1$ $(1 \leq j \leq l)$. Then, we construct the combined graph $G = (V, E)$ such that $V = V(G_1)$ and $E = \{(u, v)|(u, v \in V(G_1)) \wedge (\forall j : 1 \leq j \leq l, (f_{i_j}(u), f_{i_j}(v)) \in E(G_{i_j}))\}$. It can be shown that, with the revision, Lemma 3.3 holds.

4. EXPERIMENTAL RESULTS

We conducted an extensive performance study using both real data sets and synthetic data sets. The algorithms were implemented in Java and the experiments were run on a Sun Ultra 10 work station with a 440MHz CPU and 256 MB main memory.

4.1 The Data Sets

4.1.1 The Genomic Data Set

We used a real data set consisting of the gene expression data CDC28 and the protein-protein interaction data DIP.

The CDC28 data set [8] records the mRNA transcript levels of the budding yeast *S. cerevisiae* during the cell cycle. It contains the expression values of $6,096$ ORFs (genes) during a 17-point time-series, and is publicly available at http://cellcycle-www.stanford.edu.

The protein-protein interaction data DIP, which is publicly available at http://dip.doe-mbi.ucla.edu, is a database of interacting proteins. We downloaded the version on June 6th, 2004, of the *S. cerevisiae* subset (yeast20040606.lst). This data set contains $15,409$ pairs of interacting proteins identified in the yeast *S. cerevisiae*.

We found $4,668$ matched gene-protein pairs between CDC28 and DIP. For CDC28 data set, we used the Pearson's correlation coefficient as the measure of coherence and set the coherence threshold $\rho = 0.5$. As the result, the gene graph G_E contains $865,080$ edges whose both endpoints (genes) appear in the matched gene-protein pairs. After removing the self-interacting protein pairs, the protein graph G_P contains $15,115$ edges whose both endpoints (proteins) appear in the matched gene-protein pairs.

In our experiments, we found the complete set of quasi-cliques across the gene graph G_E and the protein graph G_P. Unless particularly specified, we set $\gamma_E = 1$ for G_E, $\gamma_P = 0.5$ for G_P, and $min_s = 5$.

4.1.2 Synthetic Data Sets

We wrote a data generator for synthetic data sets, which generates synthetic data sets as follows.

Given the number of graphs k and a set of vertices V, the data generator first creates k graphs $G_1, \ldots, G_k$ such that for each graph G_i, $V(G_i) = V$ and $E(G_i) = \emptyset$. Then, given the expected number of cross-graph quasi-cliques N_q and the parameters $\gamma_1, \ldots, \gamma_k$, the data generator randomly generates N_q cross-graph quasi-cliques and embeds them into graphs $G_1, \ldots, G_k$. The size of the cross-graph quasi-cliques is uniformly distributed between $qMin$ and $qMax$ that are specified by user. Finally, given the density value σ_i for graph G_i, the data generator keeps adding randomly generated edges into the graph G_i until the overall density of G_i reaches σ_i. Here, the density of a graph G is defined as

$$density(G) = \frac{|E(G)|}{\frac{(|V(G)| \cdot (|V(G)| - 1))}{2}} = \frac{2|E(G)|}{(|V(G)| \cdot (|V(G)| - 1))}.$$

In the experiments reported in this section, the default values for the parameters were as follows: $k = 2$, $\gamma_1 = 1$ for G_1, $\gamma_2 = 0.5$ for G_2, $min_s = 5$, $qMin = 5$ and $qMax = 20$.

The experimental results on real data sets and synthetic data sets are consistent. Limited by space, we use the results from the real data set to illustrate the effectiveness of the mining and the effect of the pruning techniques, and use both the real data set and the synthetic data sets to examine the efficiency and the scalability.

4.2 Results on the Genomic Data

We mined cross-graph quasi-cliques from the genomic data set. Figure 10 shows an example cross-graph quasi-clique Q ($\gamma_E = 1$ and $\gamma_P = 0.4$), where the diameter is only 3. The induced graph of G_E (the gene expression graph) on Q is a perfect clique, so we only show the induced graph of G_P (the protein interaction graph) on Q here. The cross-graph quasi-clique contains 11 vertices. We use the ORF (Open Reading Frame) names to represent the corresponding genes and proteins.

The cross-graph quasi-clique is interesting in biology since these 11 genes are highly coherent and the corresponding 11 proteins are intensively interacting.

To compare the efficiency of the rudimentary algorithm (Section 3.1) and *Crochet*, we got subsets of genes and proteins in the genomic data set by sampling. We used the induced graphs on the samples to test both algorithms. The number of vertices in the sub-

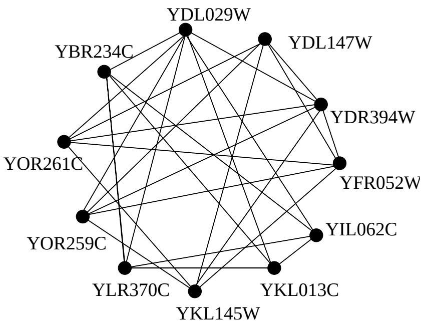
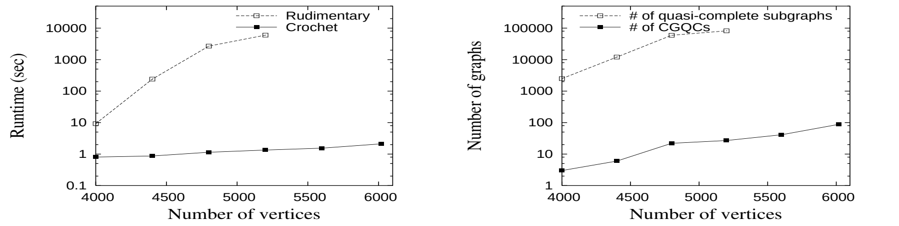


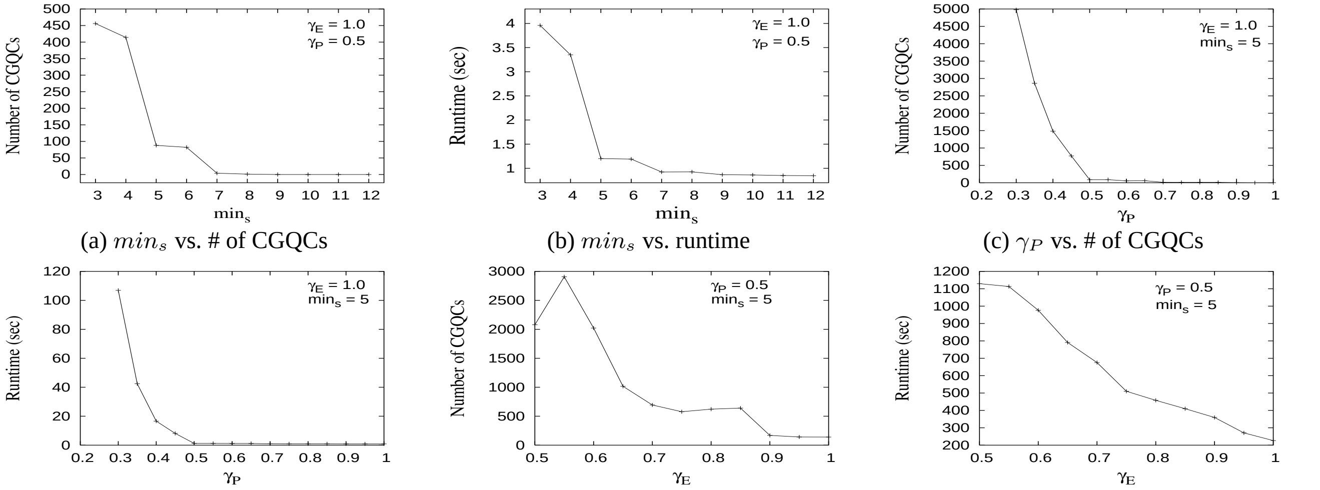
Figure 10: A cross-graph quasi-clique of 11 **proteins.**



(a) Runtime

(b) # of quasi-complete subgraphs vs. CGQCs

Figure 11: Comparison between the rudimentary algorithm and *Crochet* ($\gamma_E = 1$, $\gamma_P = 0.5$ and $min_s = 5$).



(a) min_s vs. # of CGQCs

(b) min_s vs. runtime

(c) γ_P vs. # of CGQCs

(d) γ_P vs. runtime

(e) γ_E vs. # of CGQCs

(f) γ_E vs. runtime

Figure 12: Effects of min_s, γ_P and γ_E.

sets ranges from $4,000$ to $6,018$ (the whole data set). The runtime is shown in Figure 11(a). Please note that axis Y is in logarithmic scale. Clearly, *Crochet* is dramatically more efficient than the rudimentary method. This strongly indicates that the techniques in *Crochet* are effective.

To understand the huge difference between the efficiency of the two algorithms, recall that the rudimentary algorithm computes the complete set of quasi-complete subgraphs in G_E. In Figure 11(b), we plot the number of quasi-complete subgraphs in G_E and the number of cross-graph quasi-cliques, respectively. Clearly, the number of cross-graph quasi-cliques is much smaller. That partly explains the saving in *Crochet*. Since *Crochet* is dramatically faster and more scalable than the rudimentary algorithm, hereafter, we focus on analyzing the performance of *Crochet*.

Figures 12 (a) and (b) show the number of cross-graph quasi-cliques and the runtime of *Crochet* with respect to parameter min_s, the minimum number of vertices in a cross-graph quasi-clique. As can be seen, when min_s is large, the number of cross-graph quasi-cliques is small and the runtime is short. As min_s decreases, the number of cross-graph quasi-cliques may increase substantially and the runtime also increases accordingly. The two curves follow the same trend.

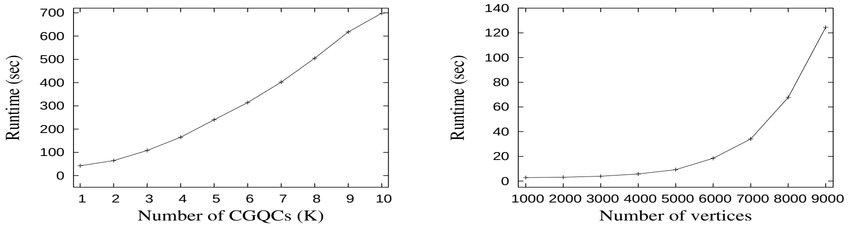
We also tested the effect of γ_P and γ_E, as shown in Figures 12(c)-(f). We fixed $\gamma_E = 1$ in Figures 12(c) and (d). The two curves follow the same trend. When $\gamma_P \geq 0.5$, the cross-graph quasi-cliques are compact and the number of cross-graph quasi-cliques is small. When $\gamma_P < 0.5$, the number of cross-graph quasi-cliques increases substantially and so does the runtime.

Technique	Runtime w/o the tech.	Runtime w/ the tech.	Speedup
Graph reduction & projection	27.819	16.392	1.697
Heuristic 1	44.274	16.392	2.701
Rule (1), Lemma 3.6	16.765	16.392	1.023
Rule (2), Lemma 3.6	130.460	16.392	7.959
Rule (3), Lemma 3.6	10.838	16.392	0.661
Rule (4), Lemma 3.6	338.916	16.392	20.676

Figure 13: The effect of various techniques in *Crochet* ($\gamma_E = 1.0$, $\gamma_P = 0.5$, $min_s = 5$).

In Figures 12(e) and (f), we fixed $\gamma_P = 0.5$ and varied γ_E from 0.5 to 1. Interestingly, the number of cross-graph quasi-cliques does not decrease monotonically when γ_E increases. The reason is that when γ_E increases, some large cross-graph quasi-cliques may split into several smaller ones. The runtime of *Crochet* decreases consistently as γ_E increases.

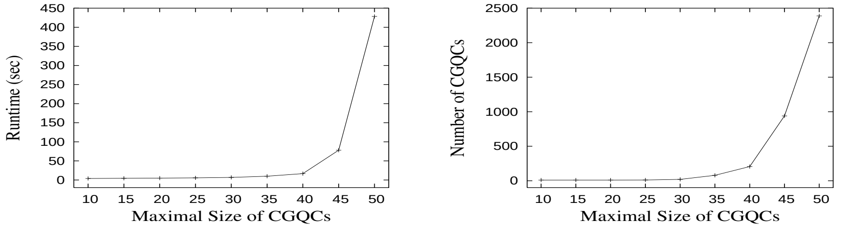
To examine the effect of the techniques in *Crochet* on improving the efficiency, we tested the speedup of the specific techniques. That is, for a specific technique, we recorded the ratio of the runtime of *Crochet* without the technique against using the technique.



(a) w.r.t. # CGQCs

(b) w.r.t. # vertices

Figure 14: Scalability on synthetic data sets.



(a) Runtime

(b) Number of CGQCs

Figure 15: Scalability with respect to the maximal size of cross-graph quasi-cliques.

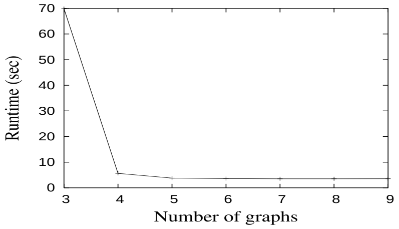


Figure 16: Scalability with respect to the number of graphs.

The results are shown in Figure 13. From the figure, we can see that rules (2) and (4) in Lemma 3.6 are most effective in pruning. Interestingly, rule (3) in Lemma 3.6, checking whether a potential quasi-complete subgraph is a subset of a cross-graph quasi-clique, is expensive and costs more than it can save. However, to avoid duplicates in output, this step cannot be removed from the algorithm.

4.3 Scalability Results on Synthetic Data Sets

Using synthetic data sets, we tested the scalability of *Crochet* on five aspects, namely (1)the number of cross-graph quasi-cliques in the data set; (2) the number of vertices in the graphs; (3) the density of the graphs; (4) the maximal size of the cross-graph quasi-cliques; and (5) the number of graphs. Among the five factors, we observed that the number of cross-graph quasi-cliques is the determinant one. The reason is that the more cross-graph quasi-cliques in the graphs, the fewer nodes we can prune from the set enumerate tree, and the longer the runtime. However, the other factors may also influence the runtime. For example, the more edges in the graphs, more cross-graph quasi-cliques may exist.

We created synthetic graphs with 1,000 vertices and embedded 1,000-10,000 cross-graph quasi-cliques in the graphs. Figure 14(a) illustrates the runtime with respect to the number of cross-graph quasi-cliques. With the graph reduction, graph projection and other pruning rules, we can see that *Crochet* is approximately linearly scalable with respect to the number of cross-graph quasi-cliques.

We also tested the runtime of *Crochet* with respect to the number of vertices (Figure 14(b)). We fixed the density of both graphs at 2% and embedded 100 expected cross-graph quasi-cliques in the graphs. As the number of vertices increases, the number of cross-graph quasi-cliques as well as their sizes increase substantially. That leads to the rapid increase of the runtime.

We explored the scalability of *Crochet* with respect to the density of the graphs. The general trend is that the runtime of *Crochet* increases as the density increases. Moreover, the increase of density in the graph with higher γ value brings more significant effect to the increase of *Crochet* runtime. This is because, likely, the number of cross-graph quasi-cliques is bounded by the graphs with higher γ values. When the γ values are set to be the same, then the increase of density in the graph with the lowest density brings the most significant effect on the runtime. Please note that, in our synthetic data sets, the noisy edges are randomly distributed to satisfy the density requirement. In real applications, the distribution of edges may not be even. Limited by space, we omit the details here.

Figure 15 shows the runtime of *Crochet* and the number of cross-graph quasi-cliques with respect to the maximal size of cross-graph quasi-cliques, respectively. As can be seen, the larger the embedded cross-graph quasi-cliques, the more skewed the edges are distributed. Thus, the number of cross-graph quasi-cliques and the runtime increase substantially.

Last, Figure 16 shows the scalability of *Crochet* with respect to the number of graphs. We fixed the number of vertices to $1,000$ and generated multiple graphs with density 10%. Since the more graphs, the less likely and the smaller cross-graph quasi-cliques are formed, and also the sharper the cross graph pruning (edge reduction and vertex reduction) is, the runtime of *Crochet* drops dramatically when the number of graphs increases.

From the above experiments, we can see that *Crochet* is efficient and scalable in mining large graphs. On the other hand, scalable mining of dense graphs and graphs with large cross-graph quasi-cliques is still challenging.

5. RELATED WORK

To the best of our knowledge, [1] and [21] are the two previous studies most related to this paper. In [1], Abello et al. defined a γ-clique in a graph G as a subset of vertices $S \subseteq V(G)$ such that the induced graph on S is connected and $|E(G(S))| \geq \gamma \cdot \binom{|S|}{2}$. They also proposed a greedy randomized adaptive search algorithm, *GRASP*, to find a γ-clique with the maximum size. In [21], Matsuda et al. introduced a definition of quasi-clique similar to ours in this paper. However, instead of finding the complete set of quasi-cliques in the graph, they proposed an approximation algorithm to cover all the vertices in the graph G with a minimum number of p-quasi-complete subgraphs. However, both [1] and [21] *neither find the complete set of quasi-cliques, nor address mining multiple graphs*.

To a more general extent, graph mining has become an important topic in data mining. For example, mining frequent substructures and subgraph patterns from many graphs (i.e., a graph database) has been studied intensively [4, 27, 16, 17, 20, 32, 31, 28]. Yan et al. [33] used frequent graph patterns to index graphs. Frequent graph pattern mining in those previous studies focuses on finding the common embedded subgraphs that appear in many graphs, which is very different from the problem of mining cross-graph quasi-cliques investigated in this study. For a cross-graph quasi-clique, the induced graphs on the clique can be very different from graph to graph. Therefore, those frequent graph pattern mining algorithms cannot be extended to mine cross-graph quasi-cliques.

In addition to frequent graph pattern mining, Palmer et al. [23] developed a fast and scalable tool *ANF* to answer various complex analytical queries from massive graphs that may not be able to fit into main memory. Faloutsos et al. [13] investigated the problem of fast discovery of connection subgraphs which nicely capture the relationship between pairs of nodes in large social networks graphs. In [18], Jeh and Widom proposed the problem of mining the space of graph properties.

Besides graph mining, graph-based algorithms are applied to cluster large data sets. A data set can be modeled as a graph, and the problem of clustering can be converted to some traditional graph problems, such as finding (quasi-)cliques or minimum cut in the graph. For example, the spectral clustering approach [9] can be viewed as finding the relaxed optimal normalized cuts in a weighted graph. A spectral clustering algorithm that uses the k eigenvectors of the adjacency matrix simultaneously was presented in [22].

As another frontier of data mining research, mining from multiple sources has received more and more attention. The papers in [11] provided good examples for techniques and applications of mining multiple relational tables.

On the application side, recent technical advances have enabled collections of many different types of biological data at a genome-wide scale, such as DNA and protein sequences, gene expression measurements, and protein-protein interactions. Various clustering approaches, including the graph-based algorithms, have been developed to explore interesting patterns in those data sets. For example, Hartuv et al. [15] proposed an algorithm, *HCS*, to find groups of genes that have similar expression patterns. *HCS* recursively splits the weighted gene graph G into a set of highly connected components along the minimum cut. Each highly connected component is considered as a gene cluster. Motivated by *HCS*, Shamir et al. developed the algorithm *CLICK* [26]. In [5], Ben-dor et al. presented a heuristic algorithm *CAST* to iteratively identify the maximal cliques once at a time. Xu et al. [30] generated a *Minimum Spanning Tree* (MST) from the weighted gene graph G. By removing $(K-1)$ edges from the generated MST, the data set is partitioned into K clusters. Moreover, Enright et al. [12] developed *TRIBE-MCL* based on *MCL* [10], a graph-based algorithm using flow simulation, to efficiently detect protein families from large protein sequence databases. Bu et al. [6] used the spectral clustering method [22] to analyze the topological structure in the protein interaction graphs. Moreover, Bader et al. [2] proposed a heuristic approach, *MCODE*, based on the concept of scale-free networks [3] to find molecular complexes in large protein interaction graphs.

Recently, joint mining of multiple biological data sets has received intense interest. As a pioneer work, Segal et al. [25] proposed a unified probabilistic model to learn the pathways from gene expression data and protein interaction data. However, their method requires the users to input the number of pathways that is often unknown in advance.

6. CONCLUSIONS

In this paper, we proposed a novel and interesting problem, mining cross-graph quasi-cliques from multiple graphs, and showed some promising application examples. The complexity analysis showed that the problem is difficult. We developed an efficient algorithm, *Crochet*, which exploits several effective techniques to mine the complete set of cross-graph quasi-cliques. An extensive performance study using both real data sets and synthetic data sets illustrated that the mining results are interesting and algorithm *Crochet* is efficient and scalable.

7. REFERENCES

[1] Abello, J. et al. Massive quasi-clique detection. *LATIN*, pages 598–612, 2002.

[2] Bader, G.D. and Hogue, C.W. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1)(2), Jan. 2003.

[3] Barabási, A.L. and Albert, R. Emergence of scaling in random networks. *Science*, 286, Oct. 1999.

[4] Bayada, D.M. et al. An algorithm for the nultiple common subgraph problem. *J. of Chem Info & Comp Sci*, pages 680–685, 1992.

[5] Ben-Dor, A. et al. Clustering gene expression patterns. *J. of Comp Biology*, 6(3/4):281–297, 1999.

[6] Bu, D. et al. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research*, 31(9):2443–2450, 2003.

[7] Cheng, Y. and Church, G.M. Biclustering of expression data. *ISMB'00*.

[8] Cho, R.J. et al. A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle. *Molecular Cell*, Vol. 2(1):65–73, July 1998.

[9] Ding, C. et al. A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering. In *ICDM'01*.

[10] Dongen, S.V. Graph clustering by flow simulation. *PhD Thesis, University of Utrecht, The Netherlands*, 2000.

[11] Dzeroski, Saso and Raedt, Luc De. On Multi Relational Data Mining (MRDM). *SIGKDD Explorations*, 5(1), July 2003.

[12] Enright, A.J., Dongen, S.V. and Ouzounis, C.A. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Reserach*, 30(7):1575–1584, 2002.

[13] Faloutsos, C., McCurley, K.S. and Tomkins, A. Fast discovery of connection subgraphs. In *KDD'04*.

[14] M. Garey and D. Johnson. *Computers and Intractability: a Guide to The Theory of NP-Completeness*. Freeman and Company, New York, 1979.

[15] Hartuv, E. and Shamir, R. A clustering algorithm based on graph connectivity. *Info Processing Letters*, 76(4–6):175–181, 2000.

[16] Holder, L.B. et al. Substructure discovery in the subdue system. In *KDD'94*.

[17] Inokuchi, A. et al. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD'00*.

[18] Jeh, G. and Widom, J. Mining the Space of Graph Properties. In *KDD'04*.

[19] Kasturi, J. et al. An information theoretic approach for analyzing temporal patterns of gene expression. *Bioinformatics*, pages 449–458, 2003.

[20] Kuramochi, M. and Karypis, G. Frequent subgraph discovery. In *Proc. 2001 Int. Conf. Data Mining (ICDM'01)*, pages 313–320, San Jose, CA, Nov. 2001.

[21] Matsuda, M. et al. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theor. Comput. Sci.*, 210(2):305–325, 1999.

[22] Ng, A.Y. et al. On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14, 2001.

[23] Palmer, C. et al. ANF: A fast and scalable tool for data mining in massive graphs. In *KDD'02*.

[24] Rymon, R. Search through systematic set enumeration. In *KR'92*.

[25] Segal, E. et al. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19:i264–i272, 2003.

[26] Shamir, R. and Sharan, R. Click: A clustering algorithm for gene expression analysis. In *ISMB'00*.

[27] Takahashi, Y. et al. Recognition of largest common fragment among a variety of chemical structures. *Analytical Sciences*, pages 23–28, 1987.

[28] Wang, C. et al. Scalable mining of large disk-based graph databases. In *KDD'04*.

[29] Wang, H. et al. Clustering by Pattern Similarity in Large Data Sets. In *SIGMOD'02*.

[30] Xu, Y. et al. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, 18:536–545, 2002.

[31] Yan, X. and Han, J. Closegraph: Mining closed frequent graph patterns. In *KDD'03*.

[32] Yan, Y. and Han, J. gspan: Graph-based substructure pattern mining. In *ICDM'02*.

[33] Yan, Y. et al. Graph indexing: A frequent structure-based approach. In *SIGMOD'04*.