Effective Pruning Techniques for Mining Quasi-Cliques*

Guimei Liu and Limsoon Wong

School of Computing, National University of Singapore, Singapore

Abstract. Many real-world datasets, such as biological networks and social networks, can be modeled as graphs. It is interesting to discover densely connected subgraphs from these graphs, as such subgraphs represent groups of objects sharing some common properties. Several algorithms have been proposed to mine quasi-cliques from undirected graphs, but they have not fully utilized the minimum degree constraint for pruning. In this paper, we propose an efficient algorithm called *Quick* to find maximal quasi-cliques from undirected graphs. The *Quick* algorithm uses several effective pruning techniques based on the degree of the vertices to prune unqualified vertices as early as possible, and these pruning techniques can be integrated into existing algorithms to improve their performance as well. Our experiment results show that *Quick* is orders of magnitude faster than previous work on mining quasi-cliques.

1 Introduction

Graphs can represent complicated relationships among objects, and they have been used to model many real-world datasets. For example, a protein-protein interaction network can be represented as a graph where each vertex represents a protein and edges represent interactions between proteins. A set of microarray data can be converted to a graph in which each vertex represents a gene and an edge between two vertices represents a strong similarity between the expression data of the two corresponding genes. Highly connected subgraphs in these graphs often have significant biological implications. They can correspond to protein complexes [1] or biologically relevant functional groups [2,3,4].

The discovery of dense subgraphs from one or multiple graphs has attracted increasing attention. Cliques are the densest form of subgraphs. A graph is a clique if there is an edge between every pair of the vertices. However, this requirement is often too restrictive given that real-world datasets are often incomplete and noisy. The concept of quasi-cliques has been proposed to relax the requirement. Different definitions have been given to quasi-cliques. Here we adopt the definition that is based on the degree of individual vertices, that is, a graph is a quasi-clique if every vertex in the graph is adjacent to at least $\lceil \gamma(n-1) \rceil$ other vertices in the graph, where γ is a number between 0 and 1 and n is the number of vertices in the graph.

* This work was supported in part by a Singapore A*STAR SERC PSF grant.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 33–49, 2008.

Given a graph, the search space of the quasi-clique mining problem is the power set of its vertex set. How to efficiently and effectively prune the search space is critical to the performance of a quasi-clique mining algorithm. However, the downward closure property no longer holds on quasi-cliques, which makes mining quasi-cliques much more challenging than mining cliques. Existing algorithms for mining quasi-cliques from a single graph all use heuristic or randomized methods and they do not produce the complete set of quasi-cliques [5,6,2]. Although existing algorithms for mining quasi-cliques from a set of graphs generate the complete result, they have not fully exploited the pruning power of the minimum degree constraint [7,8].

In this paper, we propose an efficient algorithm called Quick to mine quasi-cliques, which uses several effective pruning techniques based on the degree of the vertices. These pruning techniques can effectively remove unpromising vertices as early as possible. We conducted a set of experiments to demonstrate the effectiveness of the proposed pruning techniques.

The rest of the paper is organized as follows. Section 2 gives the formal problem definition. Section 3 presents the Quick algorithm, and its performance is studied in Section 4. Related work is described in Section 5. Finally, Section 6 concludes the paper.

2 Problem Definition

In this section, we formally define the quasi-clique mining problem. We consider simple graphs only, that is, undirected graphs that have no self-loops and multi-edges. Graph isomorphism test is very complicated and costly. To simplify the problem and the presentation, we restrict our discussion to relational graphs where every vertex has a unique label. In this case, graph isomorphism test can be performed by simply comparing the vertex set and edge set of two graphs. Note that the techniques described in this paper can be applied to non-relational graphs as well. In the rest of this paper, the term "graph" refers to simple relational graphs unless otherwise stated.

A *simple graph* G is defined as a pair (V, E), where V is a set of vertices, and E is a set of edges between the vertices. Two vertices are *adjacent* if there is an edge between them. The *adjacency list* of a vertex v in G, denoted as $N^G(v)$, is defined as $\{u|(u,v) \in E\}$. The *degree* of a vertex v in G, denoted as $deg^G(v)$, is defined as $|N^G(v)|$. The adjacency list of a vertex set X, denoted as $N^G(X)$, is defined as $\{u|\forall v \in X, (u,v) \in E\}$.

The distance between two vertices u and v in a graph $G = (V, E)$, denoted as $dist^G(u,v)$, is defined as the number of edges on the shortest path between u and v. Trivially, $dist^G(u,u) = 0$, and $dist^G(u,v) = 1$ if $u \neq v$ and $(u,v) \in E$. We denote the set of vertices that are within a distance of k from vertex v as $N_k^G(v) = \{u|dist^G(u,v) \leq k\}$. The diameter of a graph G, denoted as $diam(G)$, is defined as $max_{u,v \in V} dist^G(u,v)$. A graph is called *connected* if $dist^G(u,v) < \infty$ for any $u, v \in V$.

Definition 1 (γ-quasi-clique). *A graph $G = (V, E)$ is a γ-quasi-clique ($0 \leq \gamma \leq 1$) if G is connected, and for every vertex $v \in V$, $deg^G(v) \geq \lceil \gamma \cdot (|V| - 1) \rceil$.*

According to the definition, a quasi-clique is a graph satisfying a user-specified minimum vertex degree bound, and we call γ the *minimum degree threshold*. A clique is a special case of quasi-clique with γ=1. Figure 1 shows two example quasi-cliques. Graph G_1 is a 0.5-quasi-clique, but it is not a 0.6-quasi-clique because the degree of every vertex in G_1 is 2, and 2 is smaller than $\lceil 0.6 \cdot (5 - 1) \rceil$. Graph G_2 is a 0.6-quasi-clique.

Given a graph $G = (V, E)$, graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. Graph G is called a *supergraph* of G'. If $V' \subset V$ and $E' \subseteq E$, or $E' \subset E$ and $V' \subseteq V$, then G' is called a *proper subgraph* of G, and G is called a *proper supergraph* of G'. A subgraph G' of a graph G is called an *induced subgraph* of G if, for any pair of vertices u and v of G', (u, v) is an edge of G' if and only if (u, v) is an edge of G. We also use $G(X)$ to denote the subgraph of G induced on a vertex set $X \subseteq V$. Given a minimum degree threshold γ, if a graph is a γ-quasi-clique, then its subgraphs usually become uninteresting even if they are also γ-quasi-cliques. In this paper, we mine only maximal quasi-cliques.

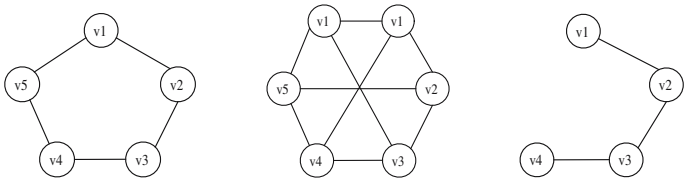
Definition 2 (Maximal γ-quasi-clique). *Given graph $G = (V, E)$ and a vertex set $X \subseteq V$. $G(X)$ is a maximal γ-quasi-clique of G if $G(X)$ is a γ-quasi-clique, and there does not exist another vertex set Y such that $Y \supset X$ and $G(Y)$ is a γ-quasi-clique.*

Cliques have the downward-closure property, that is, if G is a clique, then all of its induced subgraphs must also be cliques. This downward-closure property has been used to mine various frequent patterns in the data mining community. Unfortunately, this property does not hold for quasi-cliques. An induced subgraph of a γ-quasi-clique may not be a γ-quasi-clique. For example, graph G_1 in Figure 1 is a 0.5-quasi-clique, but one of its induced subgraph is not a 0.5-quasi-clique as shown in Figure 1(c). In fact, none of the induced subgraphs of G_1 with four vertices is a 0.5-quasi-clique.

Small quasi-cliques are usually trivial and not interesting. For example, a single vertex itself is a quasi-clique for any γ. We use a minimum size threshold *min_size* to filter small quasi-cliques.

Problem statement (Mining maximal quasi-cliques from a single graph). Given a graph $G = (V, E)$, a minimum degree threshold $\gamma \in [0, 1]$ and a minimum size threshold *min_size*, the problem of mining maximal quasi-cliques from G is to find all the vertex sets X such that $G(X)$ is a maximal γ-quasi-cliques of G and X contains at least *min_size* vertices.

In some applications, users are interested in finding quasi-cliques that occur frequently in a set of graphs. The techniques proposed in this paper can be applied to mine frequent quasi-cliques (or so-called cross quasi-cliques [7] or coherent quasi-cliques [8]) from a given graph database as well.



(a) G_1 (b) G_2 (c) $G_1(V - \{v_5\})$

Fig. 1. Examples of quasi-cliques

3 Efficient Mining of Quasi-Cliques

3.1 The Depth-First Search Framework

Given a graph $G = (V, E)$, any subset of V may form a quasi-clique. Therefore, the search space of the maximal quasi-clique mining problem is the power set of V, and it can be represented as a set-enumeration tree [9]. Figure 2 shows the search space tree for a graph G with four vertices $\{a, b, c, d\}$. Each node in the tree represents a vertex set. For every vertex set X in the tree, only vertices after the last vertex of X can be used to extend X. This set of vertices are called *candidate extensions* of X, denoted as $cand\_exts(X)$. For example, in the search space tree shown in Figure 2, vertices are sorted into lexicographic order, so vertex d is in $cand\_exts(\{a, c\})$, but vertex b is not a candidate extension of $\{a, c\}$ because vertex b is before vertex c in lexicographic order.

The Quick algorithm uses the depth-first order to explore the search space. In the example search space tree shown in Figure 2, the Quick algorithm first finds all the quasi-cliques containing vertex a, and then finds all the quasi-cliques containing vertex b but not containing vertex a, and so on. The size of the search space is exponential to the number of vertices in the graph. The main issue in mining quasi-cliques is how to effectively and efficiently prune the search space. As discussed in Section 2, quasi-cliques do not have the downward-closure property, hence we cannot use the downward-closure property to prune the search space here. According to the definition of quasi-cliques, there is a minimum

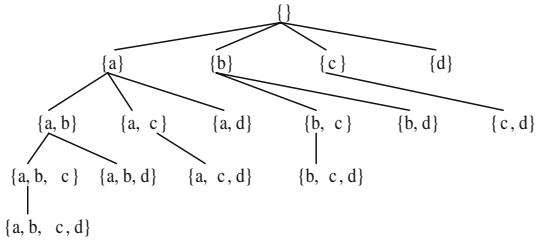


Fig. 2. The search space tree ($V = \{a, b, c, d\}$)

requirement on the degree of the vertices in a quasi-clique. We use this constraint to reduce the candidate extensions of each vertex set X.

3.2 Pruning Techniques Used in Existing Work

Before describing the new pruning techniques used in the Quick algorithm, we first describe the pruning techniques used by existing work. These pruning techniques are adopted in the Quick algorithm.

Pruning based on graph diameters. Pei et al. inferred the upper bound of the diameter of a γ-quasi-clique based on the value of γ (Theorem 1 in [7]). In particular, the upper bound of the diameter of a γ-quasi-clique is 2 when $\gamma \geq 0.5$. They used this upper bound to reduce the candidate extensions of a vertex set as stated in the following lemma.

Lemma 1. *Given graph $G = (V, E)$ and two vertex sets $X \subset Y \subseteq V$, if $G(Y)$ is a γ-quasi-clique, then for every vertex $u \in (Y - X)$, we have $u \in \bigcap_{v \in X} N_k^G(v)$, where k is the upper bound of the diameter of a γ-quasi-clique.*

Based on the above lemma, those vertices that are not in $\bigcap_{v \in X} N_k^G(v)$ can be removed from $cand\_exts(X)$.

Pruning based on the minimum size threshold. The size of a valid γ-quasi-clique should be no less than min_size. Consequently, the degree of a vertex contained in any valid γ-quasi-clique should be no less than $\lceil \gamma \cdot (min\_size - 1) \rceil$. Those vertices whose degree is less than $\lceil \gamma \cdot (min\_size - 1) \rceil$ can be removed since no valid γ-quasi-cliques contain them. Pei et al. [7] used this pruning technique in their algorithm.

Pruning based on the degree of the vertices. For a vertex set X in the search space, the Cocain algorithm proposed by Zeng et al. [10] prunes the candidate extensions of X based on the number of their neighbors in X and $cand\_ext(X)$. Given a vertex u, we use $indeg^X(u)$ to denote the number of vertices in X that are adjacent to u, and $exdeg^X(u)$ to denote the number of vertices in $cand\_exts(X)$ that are adjacent to u, that is, $indeg^X(u) = |\{v|(u, v) \in E, v \in X\}|$ and $exdeg^X(u) = |\{v|(u, v) \in E, v \in cand\_ext(X)\}|$. The Cocain algorithm uses the following lemmas to prune the search space and interested readers may refer to [10] for their proof.

Lemma 2. *If $m + u < \lceil \gamma \cdot (k + u) \rceil$, where $m, u, k \geq 0$, then $\forall i \in [0, u]$, $m + i < \lceil \gamma \cdot (k + i) \rceil$.*

Lemma 3. *Given a vertex set X and a vertex $u \in cand\_exts(X)$, if $indeg^X(u) + exdeg^X(u) < \lceil \gamma \cdot (|X| + exdeg^X(u)) \rceil$, then there does not exist a vertex set Y such that $(X \cup \{u\}) \subseteq Y \subseteq (X \cup cand\_exts(X))$, and $G(Y)$ is a γ-quasi-clique.*

For a vertex $v \in cand\_exts(X)$, if there does not exist a vertex set Y such that $(X \cup \{u\}) \subseteq Y \subseteq (X \cup cand\_exts(X))$ and $G(Y)$ is a γ-quasi-clique, then v is called an *invalid candidate extension* of X. The Cocain algorithm removes those invalid candidate extensions of X based on Lemma 3. Due to the removal of these invalid candidate extensions, some other candidate extensions of X that appear to be valid originally may become invalid apparently. Cocain does the pruning iteratively until no vertex can be removed from $cand\_exts(X)$. However, not all the invalid candidate extensions can be removed using Lemma 2.

The Cocain algorithm also checks the extensibility of the vertices in X using the following lemma.

Lemma 4. *Given vertex set X and a vertex $v \in X$, if $indeg^X(v) < \lceil \gamma \cdot |X| \rceil$ and $exdeg^X(v) = 0$, or $indeg^X(v) + exdeg^X(v) < \lceil \gamma \cdot (|X| - 1 + exdeg^X(v)) \rceil$, then there does not exist a vertex set Y such that $X \subset Y \subseteq (X \cup cand\_exts(X))$ and $G(Y)$ is a γ-quasi-clique. Vertex v is called a* failed vertex *of X.*

If there is a failed vertex in X, then there is no need to extend X further.

3.3 New Pruning Techniques Used in the Quick Algorithm

The above pruning techniques can effectively prune the search space, but they have not fully utilized the pruning power of the minimum degree constraint yet, and not all the invalid candidate extensions can be detected and removed by them. Next we describe the new pruning techniques used in our Quick algorithm.

Technique 1: pruning based on the upper bound of the number of vertices that can be added to X concurrently to form a γ-quasi-clique. Given vertex set X, the maximum number of vertices that can be added to X to form a γ-quasi-clique is bounded by the minimal degree of the vertices in X.

Lemma 5. *Let $deg_{min}(X) = \min\{indeg^X(v) + exdeg^X(v) | v \in X\}$, Y be a superset of X such that $Y \subseteq (X \cup cand\_exts(X))$ and $G(Y)$ is a γ-quasi-clique. We have $|Y| \leq \lfloor deg_{min}(X)/\gamma \rfloor + 1$.*

Proof. For every vertex $v \in X$, we have $indeg^X(v) + exdeg^X(v) \geq indeg^Y(v) \geq \lceil \gamma \cdot (|Y| - 1) \rceil$, so we have $deg_{min}(X) \geq \lceil \gamma \cdot (|Y| - 1) \rceil$. Therefore, we have $\lfloor deg_{min}(X)/\gamma \rfloor \geq \lfloor (\lceil \gamma \cdot (|Y| - 1) \rceil)/\gamma \rfloor \geq \lfloor \gamma \cdot (|Y| - 1)/\gamma \rfloor = |Y| - 1$. So we have $|Y| \leq \lfloor deg_{min}(X)/\gamma \rfloor + 1$.

Based on Lemma 5, we derive the following upper bound:

Definition 3 (U_X^{min}). *The maximal number of vertices in $cand\_exts(X)$ that can be added to X concurrently to form a γ-quasi-clique should be no larger than $\lfloor deg_{min}(X)/\gamma \rfloor + 1 - |X|$, where $deg_{min}(X) = \min\{indeg^X(v) + exdeg^X(v) | v \in X\}$. We denote this upper bound as $U_X^{min} = \lfloor deg_{min}(X)/\gamma \rfloor + 1 - |X|$.*

We further tighten this lower bound based on the observation that if $G(Y)$ is a γ-quasi-clique, then for any subset X of Y, we have $\sum_{v \in X} indeg^Y(v) \geq |X| \cdot \lceil \gamma \cdot (|Y| - 1) \rceil$.

Lemma 6. *Let vertices in cand_exts(X) be sorted in descending order of their indeg^X value, and the set of sorted vertices be denoted as $\{v_1, v_2, \cdots, v_n\}$. Given an integer $1 \leq k \leq n$, if $\sum_{v \in X} indeg^X(v) + \sum_{1 \leq i \leq k} indeg^X(v_k) < |X| \cdot \lceil \gamma \cdot (|X| + k - 1) \rceil$, then for every vertex set Z such that $Z \subseteq cand\_exts(X)$ and $|Z| = k$, $X \cup Z$ is not a γ-quasi-clique.*

Proof. Given a vertex set Z such that $Z \subseteq cand\_exts(X)$ and $|Z| = k$, we have $\sum_{v \in X} indeg^{X \cup Z}(v) = \sum_{v \in X} indeg^X(v) + \sum_{v \in X} indeg^Z(v) = \sum_{v \in X} indeg^X(v) + \sum_{v \in Z} indeg^X(v) \leq \sum_{v \in X} indeg^X(v) + \sum_{1 \leq i \leq |Z|} indeg^X(v_i) < |X| \cdot \lceil \gamma \cdot (|X| + |Z| - 1) \rceil$. Therefore, $X \cup Z$ is not a γ-quasi-clique.

Based on the above lemma, we tighten the upper bound as follows:

Definition 4 (Upper bound U_X). *Let $U_X = \max\{t | \sum_{v \in X} indeg^X(v) + \sum_{1 \leq i \leq t} indeg^X(v_i) \geq |X| \cdot \lceil \gamma \cdot (|X| + t - 1) \rceil, 1 \leq t \leq U_X^{min}\}$ if such t exists, otherwise $U_X = 0$. If $G(Y)$ is a γ-quasi-clique and $X \subseteq Y \subseteq (X \cup cand\_exts(X))$, then $|Y - X| \leq U_X$.*

Lemma 7. *Given a vertex set X and a vertex $u \in cand\_exts(X)$, if $indeg^X(u) + U_X - 1 < \lceil \gamma \cdot (|X| + U_X - 1) \rceil$, then there does not exist a vertex set Y such that $(X \cup \{u\}) \subseteq Y \subseteq (X \cup cand\_exts(X))$, and $G(Y)$ is a γ-quasi-clique.*

Proof. Let Y be a vertex set such that $G(Y)$ is a γ-quasi-clique and $(X \cup \{u\}) \subseteq Y \subseteq (X \cup cand\_exts(X))$. Since $u \in (Y - X)$, there are at most $|Y| - |X| - 1$ vertices in $Y - X$ that are adjacent to u, and $|Y| - |X| - 1 \leq U_X - 1$ based on the definition of U_X. Based on Lemma 2 and the fact that $indeg^X(u) + U_X - 1 < \lceil \gamma \cdot (|X| + U_X - 1) \rceil$, we have $indeg^X(u) + |Y| - |X| - 1 < \lceil \gamma \cdot (|X| + |Y| - |X| - 1) \rceil = \lceil \gamma \cdot (|Y| - 1) \rceil$. Therefore, we have $indeg^Y(u) \leq indeg^X(u) + |Y| - |X| - 1 < \lceil \gamma \cdot (|Y| - 1) \rceil$. It contradicts the assumption that $G(Y)$ is a γ-quasi-clique.

Similarly, we can get the following lemma, and its proof is similar to Lemma 7.

Lemma 8. *Given a vertex set X and a vertex $u \in X$, if $indeg^X(u) + U_X < \lceil \gamma \cdot (|X| + U_X - 1) \rceil$, then there does not exist a vertex set Y such that $X \subseteq Y \subseteq (X \cup cand\_exts(X))$, and $G(Y)$ is a γ-quasi-clique.*

For each vertex set X, its candidate extensions are pruned based on the upper bound as follows. We first check whether $U_X = 0$. If it is true, then no vertices in $cand\_exts(X)$ can be added to X to form a γ-quasi-clique. Next, we check whether there exists some vertex $u \in X$ such that $indeg^X(u) + U_X < \lceil \gamma \cdot (|X| + U_X - 1) \rceil$. If such u exists, then no γ-quasi-cliques can be generated by extending X. Otherwise, we remove the invalid candidate extensions of X identified by Lemma 7 from $cand\_exts(X)$. The removal of these invalid candidate extensions can in turn reduce the degree of other vertices in $cand\_exts(X)$, thus making other invalid vertices identifiable. The pruning is iteratively carried out until no more vertices can be removed from $cand\_exts(X)$.

Technique 2: pruning based on the lower bound of the number of vertices that can be added to X concurrently to form a γ-quasi-clique. Given a vertex set X, if there exists a vertex $u \in X$ such that $indeg^X(u) < \lceil \gamma \cdot (|X| - 1) \rceil$, then at least a certain number of vertices need to be added to X to increase the degree of u in order to form a γ-quasi-clique. We denote this lower bound as L_X^{min}, and it is defined as follows.

Definition 5 (L_X^{min}). *Let $indeg_{min}(X) = \min\{indeg^X(v) | v \in X\}$. L_X^{min} is defined as $L_X^{min} = \min\{t | indeg_{min}(X) + t \geq \lceil \gamma \cdot (|X| + t - 1) \rceil\}$.*

Again, this lower bound can be further tightened based on Lemma 6. We sort vertices in $cand\_exts(X) = \{v_1, v_2, \cdots, v_n\}$ in descending order of $indeg^X$ value.

Definition 6 (Lower bound L_X). *Let $L_X = \min\{t | \sum_{v \in X} indeg^X(v) + \sum_{1 \leq i \leq t} indeg^X(v_i) \geq |X| \cdot \lceil \gamma \cdot (|X| + t - 1) \rceil, L_X^{min} \leq t \leq n\}$ if such t exists. Otherwise $L_X = |cand\_exts(X)| + 1$. If $G(Y)$ is a γ-quasi-clique and $X \subseteq Y \subseteq (X \cup cand\_exts(X))$, then $|Y - X| \geq L_X$.*

Based on the definition of L_X, we can get the following lemmas.

Lemma 9. *Let X be a vertex set and u be a vertex in $cand\_exts(X)$. If $indeg^X(u) + exdeg^X(u) < \lceil \gamma \cdot (|X| + L_X - 1) \rceil$, then there does not exist a vertex set Y such that $(X \cup \{u\}) \subseteq Y \subseteq (X \cup cand\_exts(X))$, and $G(Y)$ is a γ-quasi-clique.*

Lemma 10. *Let X be a vertex set and u be a vertex in X. If $indeg^X(u) + exdeg^X(u) < \lceil \gamma \cdot (|X| + L_X - 1) \rceil$, then there does not exist a vertex set Y such that $X \subseteq Y \subseteq (X \cup cand\_exts(X))$, and $G(Y)$ is a γ-quasi-clique.*

For each vertex set X, its candidate extensions are pruned based on the lower bound as follows. We first check whether $L_X > U_X$. If it is true, then no need to extend X further. Next, we check whether there exists some vertex $u \in X$ such that $indeg^X(u) + exdeg^X(u) < \lceil \gamma \cdot (|X| + L_X - 1) \rceil$. If such u exists, then no γ-quasi-cliques can be generated by extending X based on Lemma 10. Otherwise, we remove the invalid candidate extensions of X identified by Lemma 9. Again the removal is carried out iteratively until no more candidate can be removed.

Note that the removal of the invalid candidate extensions based on Lemma 3, 7 and 9 may further tighten the two bounds L_X and U_X, which may in turn make more invalid candidate extensions identifiable.

Technique 3: pruning based on critical vertices. Let X be a vertex set. If there exists a vertex $v \in X$ such that $indeg^X(v) + exdeg^X(v) = \lceil \gamma \cdot (|X| + L_X - 1) \rceil$, then v is called a *critical vertex* of X.

Lemma 11. *If $v \in X$ is a critical vertex of X, then for any vertex set Y such that $X \subset Y \subseteq (X \cup cand\_exts(X))$ and $G(Y)$ is a γ-quasi-clique, we have $\{u | (u, v) \in E \wedge u \in cand\_exts(X)\} \subseteq Y$.*

Proof. Let u be a vertex such that $u \in cand\_exts(X)$ and $(u, v) \in E$. Suppose that $u \notin Y$, then we have $indeg^Y(v) < indeg^X(v) + exdeg^X(v) = \lceil \gamma \cdot (|X| + L_X - 1) \rceil \leq \lceil \gamma \cdot (|Y| - 1) \rceil$. It contradicts the fact that Y is a γ-quasi-clique.

Based on the above lemma, we can identify the critical vertices for every vertex set X. If such critical vertex exists, let it be v, then we add the vertices in $cand\_exts(X)$ that are adjacent to v to X. Let Y be the resultant vertex set. The remaining mining is performed on Y, and the cost for extending $X \cup \{u\} (u \notin Y)$ is saved.

Technique 4: pruning based on cover vertices. This pruning technique is inspired by the technique used in [11] for mining maximal cliques. Tomita et al. use the following lemma to prune non-maximal cliques.

Lemma 12. *Let X be a clique and u be a vertex in $N^G(X)$. For any vertex set Y such that $G(Y)$ is a clique and $Y \subseteq (X \cup N^G(X \cup \{u\}))$, $G(Y)$ cannot be a maximal clique.*

Proof. Vertex u is adjacent to all the vertices in X, and it is also adjacent to all the vertices in $N^G(X \cup \{u\})$. Hence u is adjacent to all the vertices in Y. In other words, $Y \cup \{u\}$ is a clique, thus clique Y is not maximal.

Based on the above lemma, Tomita et al. pick a vertex with the maximal degree from $cand\_exts(X)$. Let u be the picked vertex. When X is extended using vertices in $cand\_exts(X)$, the vertex set extended from X must contain at least one vertex that is not in $N^G(X \cup \{u\})$. In this way, the subsets of $(X \cup N^G(X \cup \{u\}))$ are pruned since they are not maximal.

Here we generalize the above lemma to quasi-cliques.

Lemma 13. *Let X be a vertex set and u be a vertex in $cand\_exts(X)$ such that $indeg^X(u) \geq \lceil \gamma \cdot |X| \rceil$. If for any vertex $v \in X$ such that $(u, v) \notin E$, we have $indeg^X(v) \geq \lceil \gamma \cdot |X| \rceil$, then for any vertex set Y such that $G(Y)$ is a γ-quasi-clique and $Y \subseteq (X \cup (cand\_exts(X) \cap N^G(u) \cap (\bigcap_{v \in X \wedge (u,v) \notin E} N^G(v))))$, $G(Y)$ cannot be a maximal γ-quasi-clique.*

Proof. Vertex set Y is a γ-quasi-clique, then for every vertex $v \in Y$, we have $indeg^Y(v) \geq \lceil \gamma \cdot (|Y| - 1) \rceil$. Let us look at vertex set $Y \cup \{u\}$. (1) Vertex u is adjacent to all the vertices in $cand\_exts(X) \cap N^G(u) \cap (\bigcap_{v \in X \wedge (u,v) \notin E} N^G(v))$ and $indeg^X(u) \geq \lceil \gamma \cdot |X| \rceil$, so we have $indeg^{Y \cup \{u\}}(u) = indeg^X(u) + |Y| - |X| \geq \lceil \gamma \cdot |X| \rceil + |Y| - |X| \geq \lceil \gamma \cdot |Y| \rceil$. (2) Similarly, for every vertex $v \in X$ such that $(u, v) \notin E$, v is adjacent to all the vertices in $cand\_exts(X) \cap N^G(u) \cap (\bigcap_{v \in X \wedge (u,v) \notin E} N^G(v))$ and $indeg^X(u) \geq \lceil \gamma \cdot |X| \rceil$, so we have $indeg^{Y \cup \{u\}}(v) = indeg^X(v) + |Y| - |X| \geq \lceil \gamma \cdot |X| \rceil + |Y| - |X| \geq \lceil \gamma \cdot |Y| \rceil$. (3) For every vertex $v \in X$ such that $(u, v) \in E$, we have $indeg^{Y \cup \{u\}}(v) = indeg^Y(v) + 1 \geq \lceil \gamma \cdot (|Y| - 1) \rceil + 1 \geq \lceil \gamma \cdot |Y| \rceil$. (4) Similarly, for every vertex $v \in (Y - X)$, we have $indeg^{Y \cup \{u\}}(v) = indeg^Y(v) + 1 \geq \lceil \gamma \cdot |Y| \rceil$. In summary, $Y \cup \{u\}$ is a γ-quasi-clique and Y is not maixmal.

We use $C_X(u)$ to denote the set of vertices that are covered by u with respect to X, that is, $C_X(u) = cand\_exts(X) \cap N^G(u) \cap (\bigcap_{v \in X \wedge (u,v) \notin E} N^G(v))$. Based on the above lemma, we find a vertex that maximize the size of $C_X(u)$ from

$cand\_exts(X)$. Let u be the picked vertex. We call u the *cover vertex* of X. We prune those vertex sets that are subsets of $X \cup C_X(u)$ by putting the vertices in $C_X(u)$ after all the other vertices in $cand\_exts(X)$ and then using the vertices in $cand\_exts(X) - C_X(u)$ to extend X.

Technique 5: the lookahead technique. This pruning technique has been used in mining maximal frequent itemsets [12]. Its basic idea is that before extending X using any vertex from $cand\_exts(X)$, we first check whether $X \cup cand\_exts(X)$ is a γ-quasi-clique. If it is, then there is no need to extend X further because all the vertex sets extended from X are subsets of $X \cup cand\_exts(X)$, thus they cannot be maximal except for $X \cup cand\_exts(X)$ itself.

Algorithm 1. Quick Algorithm

Input:
X is a vertex set
$cand\_exts$ is the valid candidate extensions of X
γ is the minimum degree threshold
min_size is the minimum size threshold
Output:
true if some superset of X can form a γ-quasi-clique, otherwise false;
Description:
1. Find the cover vertex u of X; Sort vertices in $cand\_exts(X)$ such that vertices in $C_X(u)$ are after all the other vertices;
2. $bhas\_qclq$ = false;
3. **for all** vertex $v \in cand_exts(X) - C_X(u)$ **do**
4. **if** $|X| + |cand\_exts(X)| < min\_size$ **then**
5. **return** $bhas\_qclq$;
6. **if** $G(X \cup cand\_exts(X))$ is a γ-quasi-clique **then**
7. Output $X \cup cand\_exts(X)$;
8. **return** t rue;
9. $Y = X \cup \{v\}$;
10. $cand\_exts(X) = cand\_exts(X) - \{v\}$;
11. $cand_Y = cand\_exts(X) \cap N_k^G(v)$, where k is calculated based on Theorem in [7];
12. **repeat**
13. Calculate the upper bound U_Y and lower bound L_Y of the number of vertices that can be added to Y concurrently to form a γ-quasi-clique;
14. **if** there is a critical vertex u' in Y **then**
15. $Y = Y \cup (cand_Y \cap N^G(u'))$;
16. $cand_Y = cand_Y - (cand_Y \cap N^G(u'))$;
17. update U_Y and L_Y;
18. $Z = \{v \mid indeg^Y(v) + exdeg^Y(v) < \lceil \gamma \cdot (|Y| + exdeg^Y(v) - 1) \rceil \bigvee indeg^Y(v) + U_Y < \lceil \gamma \cdot (|Y| + U_Y - 1) \rceil \bigvee indeg^Y(v) + exdeg^(Y)(v) < \lceil \gamma \cdot (|Y| + L_Y - 1) \rceil, v \in X\}$;
19. **if** Z is not empty **then**
20. $cand_Y = \{\}$;
21. $Z = \{v \mid indeg^Y(v) + exdeg^Y(v) < \lceil \gamma \cdot (|Y| + exdeg^Y(v)) \rceil \bigvee indeg^Y(v) + U_Y - 1 < \lceil \gamma \cdot (|Y| + U_Y - 1) \rceil \bigvee indeg^Y(v) + exdeg^(Y)(v) < \lceil \gamma \cdot (|Y| + L_Y - 1) \rceil, v \in cand_Y\}$;
22. $cand_Y = Cand_Y - Z$;
23. **until** $L_Y > U_Y$ OR $Z = \{\}$ OR $cand_Y = \{\}$
24. **if** $L_Y \leq U_Y$ AND $|cand_Y| > 0$ AND $|Y| + |cand_Y| \geq min\_size$ **then**
25. $bhas\_superqclq$ = Quick(Y, $cand_Y$, γ, min_size);
26. $bhas\_qclq = bhas\_qclq$ OR $bhas\_superqclq$;
27. **if** $|Y| \geq min\_size$ AND $G(Y)$ is a γ-quasi-clique AND $bhas\_superqclq$==false **then**
28. $bhas\_qclq$ = true;
29. Output Y;
30. **return** $bhas\_qclq$;

3.4 The Pseudo-codes of the Quick Algorithm

Algorithm 1 shows the pseudo-codes of the Quick algorithm. When the algorithm is first called on a graph $G = (V, E)$, X is set to the empty set, and $cand\_exts(X)$ is set to $\{v | exdeg^X(v) \geq \lceil \gamma \cdot (min\_size - 1) \rceil, v \in V\}$.

The Quick algorithm explores the search space in depth-first order. For a vertex set X in the search space, Quick first finds its covering vertex u, and puts the vertices in $C_X(u)$ after all the other vertices in $cand\_exts(X)$ (line 1). Only the vertices in $cand\_exts(X) - C_X(u)$ are used to extend X to prune the subsets of $X \cup C_X(u)$ based on Lemma 13 (line 3). Before using a vertex v to extend X, Quick uses the minimum size constraint (line 4-5) and the lookahead technique to prune search space. Quick checks whether $X \cup cand\_exts(X)$ is a γ-quasi-clique. If it is, then Quick outputs $X \cup cand\_exts(X)$ and skips the generation of subsets of $X \cup cand\_exts(X)$ (line 6-8).

When using a vertex v to extend X, Quick first removes those vertices that are not in $N_k^G(v)$, where k is the upper bound of the diameter of $G(X \cup cand\_exts(X))$ if $G(X \cup cand\_exts(X))$ is a γ-quasi-clique (line 11). Let $Y = X \cup \{u\}$. Next Quick calculates the upper bound U_Y and lower bound L_Y of the number of vertices that can be added to Y concurrently to form a γ-quasi-clique, and uses these two bounds to iteratively remove invalid candidate extensions of Y as follows (line 12-23). It first identifies critical vertices in Y. If there is a critical vertex u in Y, Quick adds all the vertices in $cand\_exts(Y)$ that are adjacent to u to Y based on Lemma 11 (line 14-16). Next, Quick checks the extensibility of the vertices in Y based on Lemma 4, 8 and 10 (line 18-20). Quick then prunes invalid candidate extensions based on Lemma 3, 7 and 9 (line 21-22). If $cand_Y$ is not empty after all the pruning, Quick extends Y recursively using $cand_Y$ (line 25).

The last two pruning techniques described in Section 3.3 can remove some non-maximal quasi-cliques, but they cannot remove all. To further reduce the number of non-maximal quasi-cliques generated, we check whether there is any γ-quasi-clique generated from some superset of Y, and output Y only if there is none (line 26-28). The remaining non-maximal quasi-cliques are removed in a post-processing step. We store all the vertex sets of the γ-quasi-cliques produced by Algorithm 1 in a prefix-tree. Quasi-cliques represented by internal nodes cannot be maximal. For each quasi-clique represented by a leaf node, we search for its subsets in the tree and mark them as non-maximal. At the end, the quasi-cliques represented by the leaf nodes that are not marked as non-maximal are maximal, and they are put into the final output.

Algorithm 1 prunes the search space based on the lemmas described in Section 3.2 and 3.3, so its correctness and completeness is guaranteed by the correctness of these lemmas.

4 A Performance Study

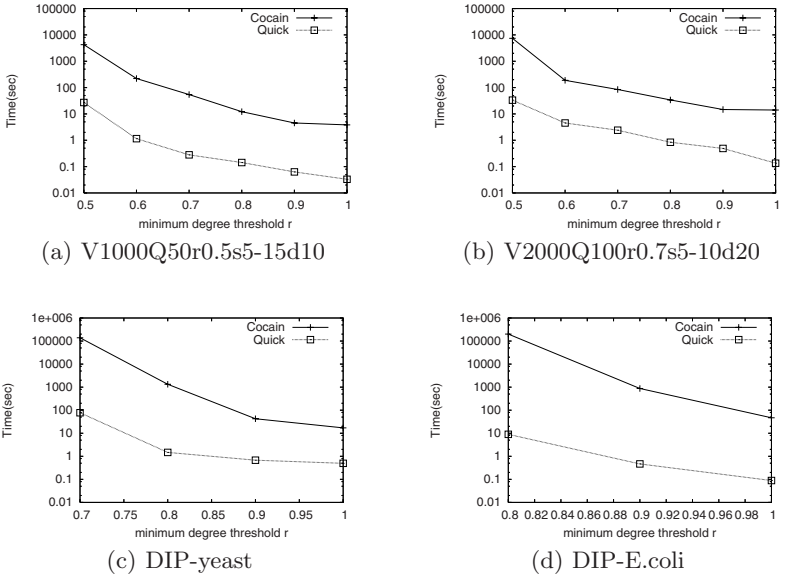
In this section, we study the efficiency of the Quick algorithm and the effectiveness of the pruning techniques used in Quick. Our experiments were conducted

on a PC with an Intel Core 2 Duo CPU (2.33GHz) and 3.2GB of RAM. The operating system is Fedora 7. Our algorithm was implemented using C++ and complied using g++.

We used both real datasets and synthetic datasets in our experiments. The real datasets are protein interaction networks downloaded from DIP(`http://dip.doe-mbi.ucla.edu/`). The yeast interaction network contains 4932 vertices (proteins) and 17201 edges (interactions). The E.coli interaction network contains 1846 vertices and 5929 edges. The synthetic graphs are generated using several parameters: V is the number of vertices, Q is the number of quasi-cliques planted in the graph, γ_{min} is the minimum degree threshold of the planted quasi-cliques, $MinSize$ and $MaxSize$ are the minimum and maximum size of the planted quasi-cliques, and d is the average degree of the vertices. To generate a synthetic graph, we first generate a value γ between γ_{min} and 1, and then generate Q γ-quasi-cliques. The size of the quasi-cliques is uniformly distributed between $MinSize$ and $MaxSize$. If the average degree of the V vertices is less than d after all the Q quasi-cliques are planted, then we randomly add edges into the graph until the average degree reaches d.

4.1 Comparison with *Cocain*

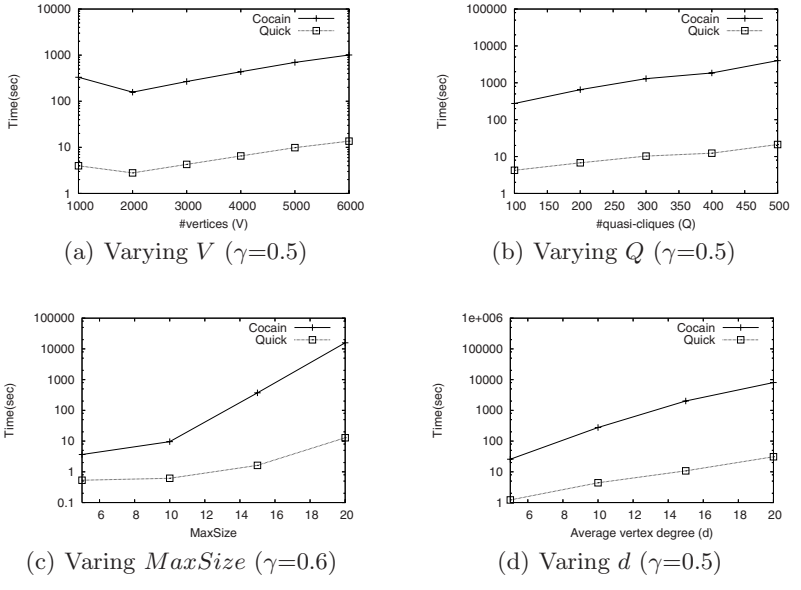
We compared Quick with Cocain [8] in terms of mining efficiency. The Cocain algorithm is designed for mining coherent closed quasi-cliques from a graph database. A quasi-clique is closed if all of its supergraphs that are quasi-cliques are less frequent than it. When applied to a single graph with minimum support



(a) V1000Q50r0.5s5-15d10 (b) V2000Q100r0.7s5-10d20

(c) DIP-yeast (d) DIP-E.coli

Fig. 3. Running time on four datasets



(a) Varying V (γ=0.5) (b) Varying Q (γ=0.5)

(c) Varing $MaxSize$ (γ=0.6) (d) Varing d (γ=0.5)

Fig. 4. Varying dataset generation parameters (default parameters: V=3000, Q=100 γ_{min}=0.5, $MinSize$=5, $MaxSize$=10, d=10.0)

of 100%, mining coherent closed quasi-cliques is equivalent to mining maximal quasi-cliques. The executable of Cocain was kindly provided by their authors.

Figure 3 shows the running time of the two algorithms with respect to the γ threshold on four datasets. The first two datasets are synthetic datasets. Dataset V1000Q50r0.5s5-15d10 was generated with V = 1000, Q=50, γ_{min}=0.5, $MinSize$=5, $MaxSize$=15 and d=10. Dataset V2000Q100r0.7s5-10d20 was generated with V = 2000, Q = 100, γ_{min}=0.7, $MinSize$=5, $MaxSize$=10 and d=20. On all four datasets, the min_size threshold is set to 1. Quick is tens of times or even hundreds of times more efficient than Cocain on all four datasets. It indicates that the pruning techniques used in Quick are very effective in pruning search space. The running time of both algorithms increases with the decrease of the γ threshold because more γ-quasi-cliques are generated and less vertices can be pruned when γ decreases.

We studied the scalability of the two algorithms using synthetic datasets. Figure 4 shows the running time of the two algorithms when varying the number of vertices V, the number of planted quasi-cliques Q, the maximum size the planted quasi-cliques $MaxSize$ and the average degree of the vertices d respectively. The default parameters are set as follows: V=3000, Q=100, γ_{min}=0.5, $MinSize$=5, $MaxSize$=10, and d=10. The running time of both algorithm increases steadily with the increase of the number of vertices and the number of planted quasi-cliques. They are more sensitive to the increase of the maximum size of the planted quasi-cliques and the average degree of the vertices. We observed the same trend on the number of maximal quasi-cliques generated.

4.2 Effectiveness of the Proposed Pruning Techniques

In this experiment, we study the effectiveness of the proposed pruning techniques. We implemented a baseline mining algorithm that does not use any of the five pruning techniques described in Section 3.3, but it uses the pruning techniques described in Section 3.2. We then add one of the five pruning techniques to the baseline algorithm. Table 1 shows the running time of the baseline algorithm with no or one of the five pruning techniques on dataset DIP-E.coli with γ=0.8 and min_size=1.

Table 1 shows that on dataset DIP-E.coli, the most effective pruning technique is the one based on the lower bound of the number of vertices that should be added to the current vertex set concurrently to form a γ-quasi-clique (Technique 2 in Section 3.3). The lookahead technique does not help very much on this dataset. However, on dataset V1000Q50r0.5s5-15d10, the lookahead technique can achieve a speedup ratio of 1.34. It implies that the effectiveness of the pruning techniques also depends on the characteristics of the datasets. The overall speedup ratio of the Quick algorithm over the baseline algorithm is 402.71, which is smaller than the multiplication of the speedup ratios of individual pruning techniques. The reason being that some invalid candidate extensions can be pruned by multiple pruning techniques.

Table 1. The effectiveness of the five pruning techniques on dataset DIP-E.coli

Algorithms	time	speedup
Baseline	3604.67	-
Baseline+UpperBound	1032.88	3.49
Baseline+LowerBound	18.73	192.48
Baseline+CriticalVertex	3572.09	1.01
Baseline+CoverVertex	2505.75	1.44
Baseline+Lookahead	3601.45	1.00
Quick	8.95	402.71

5 Related Work

The problem of determining whether a graph contains a clique of at least a given size k is a NP-complete problem [13]. It is an even harder problem to enumerate all the maximal cliques or quasi-cliques from a graph. Bron and Kerbosch [14] proposed an efficient algorithm to solve the problem more than 30 years ago, which is still one of the most efficient algorithms for enumerating all maximal cliques today. Their algorithm is recently improved by Tomita et al. [11] by using a tree-like output format.

There is a growing interest in mining quasi-cliques in recent years. Matsuda et al. [5] introduced a graph structure called p-quasi complete graph, which is the same as the γ-quasi-cliques defined in this paper, and they proposed an

approximation algorithm to cover all the vertices in a graph with a minimum number of p-quasi complete subgraphs. Abello et al. [6] defined a γ-clique in a graph as a connected induced subgraph with edge density no less than γ. They proposed a greedy randomized adaptive search algorithm called GRASP to find γ-cliques. Bu et al. [2] used the spectral clustering method to find quasi-cliques and quasi-bicliques from protein interaction networks.

The above work finds cliques or quasi-cliques from a single graph. Some work mines the clique and quasi-clique from multiple graphs. Pei et al. [7] proposed an algorithm called Crochet to mine cross quasi-cliques from a set of graphs, and they required that a quasi-clique must appear in all the graphs. The pruning techniques used in this paper is mainly based on the co-occurrences of the vertices across all the graphs. Wang et al. [15] studied the problem of mining frequent closed cliques from graph databases. A clique is frequent if it appears in sufficient number of graphs. A clique is closed if all of its super-cliques are less frequent than it. Since cliques still have the downward closure property, so mining cliques is much easier than mining quasi-cliques. Zeng et al. [8] studied a more general problem formulation, that is, mining frequent closed quasi-cliques from graph databases, and proposed an efficient algorithm called Cocain to solve the problem. The same group of authors later extended the algorithm for out-of-core mining of quasi-cliques from very large graph databases [10]. Cocain uses several pruning techniques to prune search space, but it has not fully utilized the pruning power of the minimum degree constraint yet. The pruning techniques proposed in this paper can be integrated into Crochet and Cocain to improve their performance.

There are also some work on finding densely connected subgraphs from one single graph or from a graph database. The connectivity of a subgraph can be measured by the size of its minimum cut [16,17], edge density [3] or by other measures. Hartuv and Shamir [16] proposed an algorithm called HCS which recursively splits the weighted graph into a set of highly connected components along the minimum cut. Each highly connected component is considered as a gene cluster. Yan et al. [17] investigated the problem of mining closed frequent graphs with connectivity constraints in massive relational graphs, and proposed two algorithms, CloseCut and Splat, to solve the problem. Hu et al. [3] proposed an algorithm called Codense to mine frequent coherent dense subgraphs across massive biological networks where all edges in a coherent subgraph should have sufficient support in the whole graph set. Gibson et al. [18] proposed an algorithm to find large dense bipartite subgraphs from massive graphs, and their algorithm is based on a recursive application of fingerprinting via shingles. Ucar et al. [4] used a refinement method based on neighborhoods and the biological importance of hub proteins to find dense subgraphs from protein-protein interaction networks. Bader and Hogue [1] proposed a heuristic algorithm called MCODE which is based on vertex weighting by local neighborhood density and outward traversal from a locally dense seed protein to isolate the dense regions according to given parameters.

6 Discussion and Conclusion

In this paper, we proposed several effective pruning techniques for mining quasi-cliques. These techniques can be applied to mining quasi-cliques from a single graph or a graph database. We describe the pruning techniques in the context of relational graphs where each vertex has a unique label. It is not difficult to apply these pruning techniques to non-relational graphs where different vertices may have the same label. Our preliminary experiment results show that by using these pruning techniques, our algorithm can be orders of magnitude faster than existing algorithms for the task of mining quasi-cliques from a single graph. In our future work, we will study the effectiveness of these pruning techniques for mining frequent quasi-cliques from a graph database.

References

1. Bader, G.D., Hogue, C.W.: An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4(2) (2003)
2. Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., Zhang, J., Sun, S., Ling, L., Zhang, N., Li, G., Chen, R.: Topological structure analysis of the protein interaction network in budding yeast. Nucleic Acids Research 31(9), 2443–2450 (2003)
3. Hu, H., Yan, X., Huang, Y., Han, J., Zhou, X.J.: Mining coherent dense subgraphs across massive biological networks for functional discovery. Bioinformatics 21(1), 213–221 (2005)
4. Ucar, D., Asur, S., Çatalyürek, Ü.V., Parthasarathy, S.: Improving functional modularity in protein-protein interactions graphs using hub-induced subgraphs. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 371–382. Springer, Heidelberg (2006)
5. Matsuda, H., Ishihara, T., Hashimoto, A.: Classifying molecular sequences using a linkage graph with their pairwise similarities. Theoretical Computer Science 210(2), 305–325 (1999)
6. Abello, J., Resende, M.G.C., Sudarsky, S.: Massive quasi-clique detection. In: Proc. of the 5th Latin American Symposium on Theoretical Informatics, pp. 598–612 (2002)
7. Pei, J., Jiang, D., Zhang, A.: On mining cross-graph quasi-cliques. In: Proc. of the 11th ACM SIGKDD Conference, pp. 228–238 (2005)
8. Zeng, Z., Wang, J., Zhou, L., Karypis, G.: Coherent closed quasi-clique discovery from large dense graph databases. In: Proc. of the 12th ACM SIGKDD Conference, pp. 797–802 (2006)
9. Rymon, R.: Search through systematic set enumeration. In: Proc. of the Internation Conference on Principles of Knowledge Representation and Reasoning (1992)
10. Zeng, Z., Wang, J., Zhou, L., Karypis, G.: Out-of-core coherent closed quasi-clique mining from large dense graph databases. ACM Transactions on Database Systems (TODS) 32(2), 13 (2007)
11. Tomita, E., Tanaka, A., Takahashi, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science 363(1), 28–42 (2006)
12. Bayardo Jr., R.J.: Efficiently mining long patterns from databases. In: Proc. of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 85–93 (1998)

13. Karp, R.: Reducibility among combinatorial problems. In: Proc. of a Symposium on the Complexity of Computer Computations, pp. 85–103 (1972)
14. Bron, C., Kerbosch, J.: Algorithm 457: Finding all cliques of an undirected graph. Communications of the ACM 16(9), 575–576 (1973)
15. Wang, J., Zeng, Z., Zhou, L.: Clan: An algorithm for mining closed cliques from large dense graph databases. In: Proc. of the 22nd International Conference on Data Engineering, p. 73 (2006)
16. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Information Processing Letters 76(4-6) (2000)
17. Yan, X., Zhou, X.J., Han, J.: Mining closed relational graphs with connectivity constraints. In: Proc. of the 11th ACM SIGKDD conference, pp. 324–333 (2005)
18. Gibson, D., Kumar, R., Tomkins, A.: Discovering large dense subgraphs in massive graphs. In: Proc. of the 31st International Conference on Very Large Data Bases, pp. 721–732 (2005)