Coherent Closed Quasi-Clique Discovery from Large Dense Graph Databases[*]

Zhiping Zeng[†], Jianyong Wang[†], Lizhu Zhou[†], George Karypis[‡]
Tsinghua University, Beijing, 100084, P.R.China[†]
University of Minnesota, Minneapolis, MN 55455, USA[‡]
clipse.zeng@gmail.com, {jianyong,dcszlz}@tsinghua.edu.cn, karypis@cs.umn.edu

ABSTRACT

Frequent coherent subgraphs can provide valuable knowledge about the underlying internal structure of a graph database, and mining frequently occurring coherent subgraphs from large dense graph databases has been witnessed several applications and received considerable attention in the graph mining community recently. In this paper, we study how to efficiently mine the complete set of coherent closed quasi-cliques from large dense graph databases, which is an especially challenging task due to the downward-closure property no longer holds. By fully exploring some properties of quasi-cliques, we propose several novel optimization techniques, which can prune the unpromising and redundant sub-search spaces effectively. Meanwhile, we devise an efficient closure checking scheme to facilitate the discovery of only closed quasi-cliques. We also develop a coherent closed quasi-clique mining algorithm, **Cocain**[1]. Thorough performance study shows that Cocain is very efficient and scalable for large dense graph databases.

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

General Terms: Algorithms

Keywords: Graph mining, Quasi-clique, Coherent subgraph.

1. INTRODUCTION

Recently several studies have shown that mining frequent coherent subgraphs is especially useful, where a **coherent** subgraph can be informally defined as a subgraph that satisfies a minimum cut bound (and the formal definition can be found in Section 2.1), as the set of frequent coherent subgraphs mined from a graph database usually reflects the density distribution of the relationships among the objects in the database, and can provide valuable knowledge about the internal structure of the graph database. Coherent subgraph mining has also been witnessed several applications such as highly correlated stock discovery [5], gene function annotation [3].

Several coherent subgraph discovery algorithms have been recently developed [3, 4, 6, 5]. However, each of the previously proposed algorithms has its own limitations. For example, the algorithm proposed in [4] can only mine quasi-cliques with exact 100% support threshold; the algorithms proposed in [6] can only work on relational subgraphs, where vertex labels are distinct in each graph; similarly, the CLAN algorithm [5] mines fully connected frequent subgraphs (i.e., frequent cliques). In this paper, we study a more general problem formulation: mining frequent quasi-cliques from large dense graph databases, that is, we neither limit the minimum support to be 100% nor require the input graphs to be relational.

While the problem becomes more general, it gets more tough. As we will see later, the downward-closure property[1] no longer holds, devising some effective search space pruning techniques is especially challenging. By fully exploring some properties of quasi-cliques, we propose several novel optimization techniques, meanwhile, we devise an efficient closure checking scheme in order to facilitate the discovery of only closed quasi-cliques. We also develop an efficient coherent closed quasi-clique mining algorithm, **Cocain**.

The remaining of this paper is organized as follows. Section 2 introduces the problem formulation. In Section 3, we present algorithm **Cocain** by focusing on several optimization techniques. The empirical results are examined in Section 4. Finally, we conclude our study in Section 5.

2. PROBLEM FORMULATION

In this section, we introduce some preliminary concepts, notations, and terms in order to simplify our discussion. We also formulate the problem we study. Table 1 summarizes the notations we use in this paper and their meanings.

Notations	**Description**
V	$V = \{v_1, v_2, ..., v_k\}$, the set of vertices
E	$E \subseteq V \times V$, the set of edges
L	the set of vertex labels
F	$F : V \to L$, the mapping function from labels to vertices
G	$G = (V, E, L, F)$, an undirected vertex-labeled graph
$\|G\|$	$\|G\| = \|V\|$, the cardinality of G
$G(S)$	the induced subgraph on S from G, $S \subseteq V(G)$
$N^G(v)$	$N^G(v) = \{u \| (v, u) \in E(G)\}$
$deg^G(v)$	$deg^G(v) = \|N^G(v)\|$
$dis^G(u, v)$	the number of edges in the shortest path between u and v

Table 1: Notations used in this paper

2.1 Preliminaries

In this paper, we consider **simple graph** only, which does not contain self-loops, multi-edges, and edge labels. An **undirected**

[*]This work was supported by National Natural Science Foundation of China (NSFC) under Grant No. 60573061 and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology(TNList).

[1]Cocain stands for **Co**herent **c**losed qu**a**si-cl**i**que mi**n**ing.

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...$5.00.

vertex-labeled graph transaction, G, can be represented by a 4-tuple, $G{=}(V, E, L, F)$. An **induced subgraph** of a graph G is a subset of the vertices of $V(G)$ together with any edges whose endpoints are all in this subset. In the following discussions, the term "graph" means the undirected vertex-labeled graph, unless otherwise stated.

DEFINITION 2.1. **(γ-Quasi-clique)** *A k-graph($k{\geq}1$) G is a γ-quasi-clique* $(0{\leq}\gamma{\leq}1)$ *if* $\forall v{\in}V(G)$, $deg^G(v){\geq}\lceil\gamma{\cdot}(k{-}1)\rceil$.

From the definition we can see that quasi-cliques are subgraphs that satisfy a user-specified minimum vertex degree bound, $\lceil\gamma{\cdot}(k{-}1)\rceil$. Apparently, a γ-quasi-clique must be a fully connected graph when $\gamma = 1$, and singleton graphs are considered as γ-quasi-cliques. If $\exists v{\in}V(G)$ such that $deg^G(v){=}\lceil\gamma{\cdot}(k{-}1)\rceil$ and $\lceil\gamma{\cdot}(k{-}1)\rceil{\neq}\lceil\gamma{\cdot}k\rceil$, v is called a **critical vertex** of G w.r.t γ.

Most of the existing frequent subgraph mining algorithms are based on the downward-closure property. Unfortunately, this property does not hold for quasi-clique patterns. An induced subgraph of a γ-quasi-clique may not be a γ-quasi-clique. For instance, in Figure 1, $G_2(\{v_1{,}v_3{,}v_4{,}v_5{,}u_6\})$ is a 0.5-quasi-clique, but one of its induced subgraphs, $G_2(\{v_1{,}v_3{,}v_5{,}u_6\})$ is not a 0.5-quasi-clique.

DEFINITION 2.2. **(Edge Cut and Edge Connectivity)** *Given a connected graph $G = (V, E)$, an edge cut is a set of edges E_c such that $G'{=}(V{,}E{-}E_c)$ is disconnected. A minimum cut is the smallest set among all edge cuts. The edge connectivity of G, denoted by $\kappa(G)$, is the size of the minimum cut.*

As shown in [6], the minimum vertex degree can reflect the level of connectivity to some extent in a graph, but they cannot guarantee the graph is connected in a balanced way. However, the following lemma gives a lower bound on the minimum cut when $\gamma{\geq}0.5$, which guarantees the coherency of γ-quasi-cliques.

LEMMA 2.1. **(Minimum Edge Connectivity)** *Let n-graph $Q = (V, E)$ be a γ-quasi-clique* $(0.5{\leq}\gamma{\leq}1{,}n{\geq}2)$. *The edge connectivity of Q must be no smaller than $\lfloor\frac{n}{2}\rfloor$, i.e., $\kappa(Q){\geq}\lfloor\frac{n}{2}\rfloor$.*

PROOF. Let us divide V into two sets, V_1 and V_2, and suppose $V_1 \leq V_2$ and $|V_1|{=}k$, $1{\leq}k{\leq}\lfloor\frac{n}{2}\rfloor$ must hold.

Since Q is a γ-quasi-clique, $\forall v{\in}V_1$, $deg^Q(v){\geq}\lceil\gamma{\cdot}(n{-}1)\rceil$. However, v is at most adjacent to other $k{-}1$ vertices in V_1, and $k{-}1{\leq}\frac{n}{2}{-}1{<}\ \gamma(n{-}1)$. Therefore, v must be adjacent to vertices belonging to V_2, and the number of edges which connect v and vertices in V_2 must be no smaller than $\lceil\gamma{\cdot}(n{-}1)\rceil{-}(k{-}1)$. There are k vertices in V_1, so there exist at least $k{\cdot}(\lceil\gamma{\cdot}(n{-}1)\rceil{-}(k{-}1))$ edges between V_1 and V_2. Let $f(k){=}k{\cdot}(\lceil\gamma{\cdot}(n{-}1)\rceil{-}(k{-}1))$, then

$$f(k) = -k^2 + k \cdot (\lceil\gamma \cdot (n-1)\rceil + 1) \tag{1}$$

The second-order derivative of $f(k)$ (i.e., $\frac{d^2f(k)}{dk^2}{=}{-}2$) is negative, $f(k)$ achieves the maximum at its sole stationary point $k = \frac{\lceil\gamma\cdot(n-1)\rceil+1}{2}$. As there exists only one stationary point for quadratic polynomial function $f(k)$ and $1{\leq}k \leq\lfloor\frac{n}{2}\rfloor$, $f(k)$ must get its minimum either at $k{=}1$ or at $k{=}\lfloor\frac{n}{2}\rfloor$. When $k{=}1$, $f(1){=}\lceil\gamma{\cdot}(n{-}1)\rceil{\geq}\lceil\frac{n-1}{2}\rceil{=}\lfloor\frac{n}{2}\rfloor$. While when $k{=}\lfloor\frac{n}{2}\rfloor$, each vertex in V_1 must be adjacent to at least one vertices in V_2, thus $f(\lfloor\frac{n}{2}\rfloor){\geq}\lfloor\frac{n}{2}\rfloor$. From the above result, we can get that $\forall k{\in}[1, \lfloor\frac{n}{2}\rfloor]$, $f(k){\geq}\lfloor\frac{n}{2}\rfloor$. According to the definition of edge connectivity, $\kappa(Q){\geq}\lfloor\frac{n}{2}\rfloor$. □

Because we are more interested in mining tightly connected subgraphs, we do not expect that the edge connectivity of a subgraph is too small in comparison with the minimum vertex degree. From Lemma 2.1, if $\gamma{\geq}0.5$ the vertices in the γ-quasi-clique are connected tightly and relatively evenly. In this paper, a γ-quasi-clique is said to be **coherent** if $\gamma{\geq}0.5$, and if not explicitly stated γ by default has a value no smaller than 0.5.

DEFINITION 2.3. **(γ-Isomorphism)** *A graph $G_1{=}\{V_1{,}E_1{,}L_1{,}F_1\}$ is γ-isomorphic to another graph $G_2 = \{V_2, E_2, L_2, F_2\}$ iff both of them are γ-quasi-cliques, $|G_1|{=}|G_2|$, and there exists a bijection $f{:}V_1{\rightarrow}V_2$ such that $\forall v{\in}V_1{,}F_1(v){=}\ F_2(f(v))$.*

According to the above definition, we know that the γ-isomorphism is quite different from the graph isomorphism in graph theory, which is defined as a bijection $f{:}V(G_1){\rightarrow}V(G_2)$ from a graph G_1 to another graph G_2, such that $(u, v){\in}E(G_1)$ iff $(f(u){,}f(v)){\in}E(G_2)$. While the γ-isomorphism between two γ-quasi-cliques does not imply an exact bijection edge mapping.

A **multiset** is defined as a bag of vertex labels in which the order is ignored, but multiplicity is explicitly significant. Let $M(G)$ indicate the multiset of G. From the γ-isomorphism definition, we can derive the following lemma directly: two γ-quasi-cliques, G_1 and G_2, are γ-isomorphic to each other iff $M(G_1){=}M(G_2)$.

For two γ-quasi-clique Q and Q', if $M(Q){\subseteq}M(Q')$, Q is called a **subquasi-clique** of Q', while Q' is called a **superquasi-clique** of Q. We use $Q{\sqsubseteq}Q'$ or $Q{\sqsubset}Q'$ (i.e. $Q{\sqsubseteq}Q'$ but $Q{\neq}Q'$) to denote the subquasi-clique or proper subquasi-clique relationship.

2.2 Problem Definition

A **graph transaction database**, D, consists of a set of input graphs, and the cardinality of D is denoted by $|D|$. Figure 1 shows an example of graph transaction database D, which consists of two input graphs, G_1 and G_2.

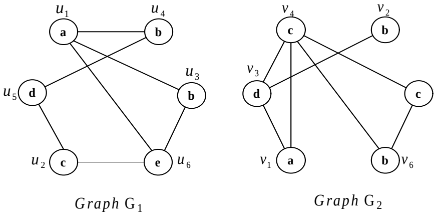


Figure 1: An example of graph database D

For two graphs G and G', let g be an induced subgraph of G, if $M(g){=}M(G')$, we call g an **instance** of G' in G. If there exists at least one instance of G' in G, we say that graph G **roughly supports** G'. Meanwhile, if g is γ-isomorphic to another γ-quasi-clique Q, we call g an **embedding** of Q in G. If there exists at least one embedding of Q in G, G is said to **strictly support** Q.

The number of input graphs in graph database D that strictly (or roughly) support a γ-quasi-clique Q (or a graph G) is called the **absolute strict-support (or absolute rough-support)** of Q (or G) in D, denoted by $sup_s^D(Q)$ (or $sup_r^D(G)$), while the **relative strict-support** $rsup_s^D(Q)$ **(or relative rough-support** $rsup_r^D(G)$**)** is defined as $rsup_s^D(Q){=}sup_s^D(Q)/|D|$(and $rsup_r^D(G){=}sup_r^D(G)/|D|$).

Given an absolute support threshold min_sup and a graph transaction database D, a quasi-clique Q (or a subgraph g) is called a **frequent** quasi-clique (or a **vice-frequent** graph) if $sup_s^D(Q) \geq min\_sup$ (or $sup_r^D(g) \geq min\_sup$). If there does not exist any another quasi-clique Q' such that $Q \sqsubset Q'$ and $sup_s^D(Q) \leq sup_s^D(Q')$, Q is called a **closed** quasi-clique in D[2].

[2] Here the definition of a 'closed' pattern is a little different from the traditional one, due to the absence of the downward-closure property for quasi-clique patterns, and it is possible that a quasi-clique's strict-support is greater than that of its subquasi-cliques. Also, in the case that $Q{\sqsubset}Q'$ and $sup_s^D(Q){\leq}sup_s^D(Q')$, we say Q' can *subsume* Q.

In this paper, given D and min_sup, we study the problem of mining the complete set of γ-quasi-cliques in D that are frequent, closed, and also coherent (i.e., $\gamma \geq 0.5$).

3. Cocain: EFFICIENTLY MINING CLOSED COHERENT QUASI-CLIQUES

In this section, we describe our comprehensive solution to frequent closed coherent quasi-clique mining, including an efficient canonical representation of a coherent quasi-clique, a subgraph enumeration framework, several search space pruning techniques, and a quasi-clique closure checking scheme. We also present the integrated algorithm, Cocain.

3.1 Canonical Representation of Subgraphs

One of the key issues in graph mining is how to choose an efficient canonical form that can uniquely represent a graph and has low computational complexity in order to facilitate the graph isomorphism testing. Some previously proposed solutions to this problem are designed for a general graph and may not be the most efficient representation for a quasi-clique. According to the Section 2.1, we can see that a multiset preserves much information for a quasi-clique. Given a k-graph g, we call any sequence of all elements in $M(g)$ a graph **string**. Assume there is a total order on the vertex labels, we define the following total order of any two strings p and q with size $|p|$ and $|q|$ respectively. Let p_i denote the i-th vertex label in string p, we define $p{<}q$ if either of the following two conditions holds: (1) $\exists t$ $(0{<}t{\leq}min\{|p|,\ |q|\})$ such that $\forall\ i\ \in[1, t-1]$, $p_i{=}q_i$ and $p_t{<}q_t$, (2) $(|p|{<}|q|)$ and $\forall\ i\ \in[1, |p|]$, $p_i{=}q_i$; otherwise $p{\geq}q$. A string $S_a{=}a_1a_2...a_n$ is called a **substring** of another string $S_b{=}b_1b_2...b_m$, denoted by $S_a{\sqsubseteq}S_b$ (or $S_a{\sqsubset}S_b$ if $m{>}n$), if there exist n integers $1{\leq}i_1{<}i_2{<}...\ i_n{\leq}m$ such that $a_1{=}b_{i_1}$,$a_2{=}b_{i_2}$,..., $a_n{=}b_{i_n}$.

DEFINITION 3.1. *The canonical form of a graph G is defined as the minimum string among all its strings and denoted by $CF(G)$.*

As we ignore the exact topology of a quasi-clique, it is evident that the above definition is a unique representation of a quasi-clique. However, we note that it does not hold for a general graph. After giving the definition of the canonical form of a graph and the substring relationship, we can derive the following two lemmas to facilitate γ-isomorphism checking and subquasi-clique relationship checking.

LEMMA 3.1. *Two γ-quasi-cliques Q_1 and Q_2 are γ-isomorphic to each other iff $CF(Q_1) = CF(Q_2)$.*

LEMMA 3.2. *Given two γ-quasi-cliques Q_1 and Q_2, $Q_1 \sqsubseteq Q_2$ (or $Q_1 \sqsubset Q_2$) iff $CF(Q_1) \sqsubseteq CF(Q_2)$ (or $CF(Q_1) \sqsubset CF(Q_2)$).*

The above two lemmas can be derived easily from the definition of γ-isomorphism and subquasi-clique relationship respectively, here we omit the proof.

3.2 Vice-Frequent Subgraph Enumeration

According to the definition of a vice-frequent graph, it is evident that any induced subgraph of a vice-frequent graph must be also vice-frequent. Thus, the downward-closure property can be exploited for vice-frequent subgraph enumeration.

Based on the definition of embedding and instance of a subgraph, we can see that the embedding is a special type of the instance, an embedding of a quasi-clique q must be an instance of q. Given a γ-quasi-clique Q, as $sup_s^D(Q){\leq}sup_r^D(Q)$ holds, if Q is frequent, Q must be vice-frequent. Consequently, we can discover the complete set of frequent γ-quasi-cliques from the set of vice-frequent subgraphs. By conceptually organizing the vice-frequent subgraphs into a lattice-like structure in the way we used in [5], the problem of mining frequent quasi-cliques becomes how to traverse the lattice-like structure to enumerate vice-frequent subgraphs and discover frequent γ-quasi-cliques.

In our running example in Figure 1, assume the total order among vertex labels is $a{\leq}b{\leq}c{\leq}d{\leq}e$, all the vice-frequent subgraphs are organized into a structure as shown in Figure 2. Note that here we use ‘canonical form:rough-support:strict-support’ to represent a subgraph. In addition, all nodes with yellow color are vice-frequent subgraphs but not frequent quasi-cliques, nodes with blue color are non-closed frequent quasi-cliques, and nodes with orange color are closed quasi-cliques. Figure 2 shows that among all the vice-frequent subgraphs, only abd:2:2 and bcd:2:2 are closed quasi-cliques. We also adopt the DFS search strategy as we used in [5] to traverse the lattice-like structure. In this way, we can get a rudimentary algorithm to discover frequent closed γ-quasi-cliques. However, due to a great deal of redundancy during the enumeration, this rudimentary algorithm is too expensive and costs too much space and runtime.

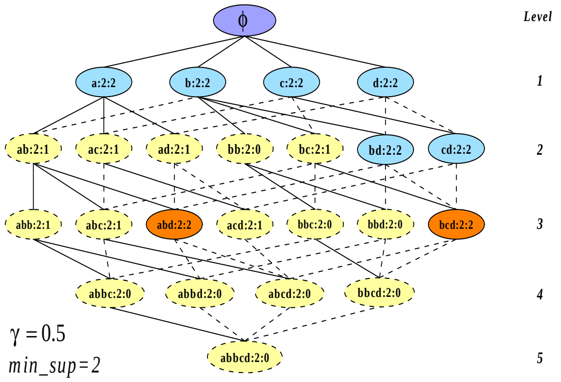


Figure 2: A lattice-like structure built from the vice-frequent subgraphs of our running example

Structural Redundancy Pruning. As shown in [5], much redundancy exists if we just simply use the DFS search strategy. In order to eliminate the structural redundancy while maintaining the completeness of the result set, we propose an efficient vice-frequent subgraph enumeration method. Given an m-graph G and $CF(G){=}\ a_0a_1...a_m$, we require G can only be generated by growing the subgraph g with canonical form $CF(g){=}a_0a_1...a_{m-1}$. In this way, except the node ϕ, each node in the lattice-like structure has only one parent, and this lattice-like structure would turn to a tree structure. Let $LAS(g)$ be the last element in $CF(g)$ (i.e., $LAS(g){=}a_m$), obviously, in the enumeration tree structure all descendants of the node g would be in the form of $CF(g) \diamond b_0b_1...b_k$, where $b_0 \geq LAS(g)$ and $\forall\ i \in [1, k]$, $b_{i-1} \leq b_i$.

3.3 Search Space Pruning

In this subsection, we propose several novel optimization techniques to prune futile search subspaces based on some nice properties of quasi-cliques.

3.3.1 Preliminaries

Before we elaborate on the pruning techniques, let us first introduce the following two important lemmas which form the foundation of several pruning techniques.

LEMMA 3.3. *If $m{+}u{<}\lceil\gamma\cdot(k+u)\rceil$ (where $m,u,k{\geq}0$, and $0.5{\leq}\gamma{\leq}1$), then $m<\lceil\gamma\cdot k\rceil$ and $\forall i{\in}[0,u]$, $m+i{<}\lceil\gamma\cdot(k+i)\rceil$.*

PROOF. First, we assume $m\geq\lceil\gamma\cdot k\rceil$, then $m+u\geq\lceil\gamma\cdot k\rceil+u\geq\lceil\gamma\cdot k\rceil+\lceil\gamma\cdot u\rceil\geq\lceil\gamma\cdot(k+u)\rceil$, which contradicts with the fact $m{+}u{<}\lceil\gamma\cdot(k+u)\rceil$. Thus, $m<\lceil\gamma\cdot k\rceil$ holds. Second, let $t{=}u{-}i$, then $m{+}i{=}m{+}u{-}t<\lceil\gamma\cdot(k+u)\rceil{-}t\leq\lceil\gamma\cdot(k+u)\rceil{-}\lceil\gamma\cdot t\rceil\leq\lceil\gamma\cdot((k+u)-t)\rceil=\lceil\gamma\cdot(k+i)\rceil$. □

LEMMA 3.4. **(Maximal Diameter)** *If a graph Q is a γ-quasi-clique, then $\forall u,v\in V(Q)$, $dis^Q(u,v)\leq 2$.*

PROOF. Let $|Q|{=}n$ $(n{\geq}2)$. $\forall u,v{\in}\ V(Q)$, if u and v are adjacent to each other, then $dis^Q(u,v){=}1$. While if u and v are not adjacent, then $|N^Q(u){\cup}N^Q(v)|{\leq}(n-2)$. Furthermore, as Q is a γ-quasi-clique, $|N^Q(u)|{\geq}\lceil\gamma{\cdot}(n{-}1)\rceil$ and $|N^Q(v)|{\geq}\lceil\gamma{\cdot}(n{-}1)\rceil$ hold. Therefore, $N^Q(u){\cap}N^Q(v){\neq}\phi$, otherwise $|N^Q(u){\cup}N^Q(v)|{=}|N^Q(u)|{+}|N^Q(v)|{\geq}2\lceil\gamma\cdot(n-1)\rceil$. Also, since $\gamma{\geq}0.5$, we have $2\lceil\gamma\cdot(n-1)\rceil{\geq}(n-1)$. Thus, $|N^Q(u){\cup}N^Q(v)|{\geq}(n-1)$ holds, which contradicts with the fact $|N^Q(u){\cup}N^Q(v)|{\leq}(n-2)$. As a result, there must exist at least one vertex which is adjacent to both u and v, and $dis^Q(u,v){=}2$ must hold. Therefore, from the above analysis, we can get $\forall u,v{\in}V(Q)$, $dis^Q(u,v){\leq}2$. □

3.3.2 Pruning Methods

From Lemma 3.4, we can derive the following lemma to help us prune some futile branches.

LEMMA 3.5. **(Diameter Pruning)** *Let G be a graph and $S\subseteq V(G)$, if $G(S)$ is a γ-quasi-clique, then $\forall$ u,v $\in S$, $dis^G(u,v){\leq}2$.*

PROOF. As $G(S)$ is a subgraph of G, $dis^{G(S)}(u,v){\geq}dis^G(u,v)$ holds. In addition, according to Lemma 3.4, we have $dis^{G(S)}(u,v)\leq 2$. Therefore, $dis^G(u,v)\leq 2$ must hold. □

Given a graph G, let $S{\subset}V(G)$ and $v{\in}V(G){-}S$, from Lemma 3.5 we know that if $G(S)$ and v can form a "bigger" quasi-clique, $\forall\, u\in S, dis^G(v,u)\leq 2$ must hold. Accordingly, a straightforward and reasonable application of Lemma 3.5 is to discover the extensible vertices of a subgraph which can be used to form quasi-cliques later. First, we calculate the extensible vertex set $E(u)$ for each vertex u in $G(S)$, then conjoin all $E(u)$'s to obtain the global extensible vertex set. Obviously, $E(u)$ can be divided into two subsets. One is denoted by $D(u)$, which consists of the vertices that are adjacent to u, i.e., $D(u)=\{v|dis^G(u,v)=1\}$; another subset is denoted by $I(u)$, where $I(u)=\{v|\exists u'\in D(u),\ dis^G(u',v)=1$ and $v\notin D(u)\}$. The set $D(u)$ can be obtained easily, and $I(u)$ can be computed from $D(u)$ in the way of discovering vertices which are adjacent to at least one vertex in $D(u)$. In this way, we can discover extensible vertex set for an instance of a subgraph $G(S)$ and can prune some unpromising vertices.

Combination Pruning. We can combine the *structural redundancy pruning* with *Diameter Pruning* to further shrink the the extensible vertex set. As stated in *structural redundancy pruning*, only those vertices whose labels are no smaller than $LAS(g)$ can be used to grow g, where g is the current prefix subgraph. Therefore, when calculating $E(u)$ we can remove the vertices whose labels are smaller than $LAS(g)$, and the removal of some vertices in $E(u)$ may make some vertices left in $E(u)$ violate the condition of an extensible vertex introduced in Lemma 3.4. For example, suppose $\exists v_0\in I(u)$ and there exists only one vertex v' such that $v'{\in}\ D(u)$ and $dis^G(v_0,v'){=}1$, if $v'\notin V(g)$ and the label of v' is smaller than $LAS(g)$, then v' will never appear in all descendants of g, thus the descendant g' of g which contains vertex v_0 cannot be a quasi-clique, as $dis^{g'}(u,v_0){>}2$ must hold. Therefore, we can remove the vertex v_0 from $E(u)$ safely. In order to eliminate these vertices efficiently, after getting $D(u)$, we remove the vertices in $D(u)$ whose labels are smaller than $LAS(g)$ and in the meantime do not belong to $V(g)$. After we compute the final set of extensible vertices, $E(u)$, for each vertex u in subgraph g, we can then conjoin all the $E(u)$'s to get the global extensible vertex set w.r.t. g. We call an element in the global extensible vertex set an **extensible candidate** w.r.t. g, and use $V^G_{cad}(g)$ to denote the set of extensible candidates w.r.t. g in G.

In the following, we propose other three optimization techniques based on Lemma 3.3 which can be used to prune the unpromising search space effectively.

For $v\in V^G_{cad}(g)$, we define the internal set $V^g_{in}(v){=}N^G(v){\cap}V(g)$ and external set $V^g_{ex}(v){=}N^G(v){\cap}V^G_{cad}(g)$. Let $indeg^g(v){=}|V^g_{in}(v)|$ be the **inner degree** of v, and $exdeg^g(v){=}|V^g_{ex}|$ be the **extern degree** of v. Take Figure 1 for an example, assume $g{=}G_2(\{v_5\})$, $V^{G_2}_{cad}(g){=}\{v_3,\ v_4\ \}$ (note v_1 and v_6 have been pruned by *Combination Pruning*), $indeg^g(v_3){=}0$, $exdeg^g(v_3){=}1$.

LEMMA 3.6. **(Vertex Connectivity Pruning)** *Suppose g is a k-subgraph of G, if $\exists\ v\in V^G_{cad}(g)$ such that $indeg(v)<\lceil\gamma\cdot k\rceil$ and $indeg^g(v)+exdeg^g(v){<}\ \lceil\gamma\cdot(k+exdeg^g(v))\rceil$, there does not exist a quasi-clique Q in G such that $V(Q)\supseteq V(g){\cup}\{v\}$.*

PROOF. Assume there exists a quasi-clique Q such that $V(Q)\supseteq V(g)\cup\{v\}$, and let $|Q|{=}l$. Since Q is a quasi-clique, $V(Q)\subseteq V(g)\cup V^G_{cad}(g)$. We define $R{=}V(Q)\cap V^g_{ex}(v)$ and denote $|R|$ by m, then $l{-}k{-}1\geq m$ and $m\leq|V^g_{ex}(v)|{=}exdeg^g(v)$. Because $indeg^g(v)<\lceil\gamma\cdot k\rceil$ and $indeg^g(v)+exdeg^g(v){<}\ \lceil\gamma\cdot(k+exdeg^g(v))\rceil$, according to the Lemma 3.3, we can get that $\forall\ i\in[0,exdeg^g(v)]$, $indeg^g(v){+}i{<}\lceil\gamma\cdot(k+i)\rceil$. Thus, $deg^Q(v){=}indeg^g(v){+}m{<}\lceil\gamma\cdot(k+m)\rceil{\leq}\ \lceil\gamma\cdot(l-1)\rceil$, i.e. $deg^Q(v){<}\lceil\gamma\cdot(|Q|-1)\rceil$. This contradicts with the assumption that Q is a quasi-clique. □

In the case of $indeg^g(v)+exdeg^g(v){<}\ \lceil\gamma{\cdot}(k{+}exdeg^g(v))\rceil$, v is called an **invalid extensible candidate**, otherwise it is called a **valid extensible candidate**. Obviously, the invalid extensible candidates do not make any contribution to the generation of "bigger" quasi-cliques. Therefore, after getting the extensible candidate set $V^G_{cad}(g)$, we could remove all invalid extensible candidates from $V^G_{cad}(g)$. Due to the removal of these vertices, some originally valid extensible candidates may turn to be invalid, so we can do this pruning iteratively until no vertex can be removed from $V^G_{cad}(g)$. We denote the remaining set by $V^G_{vad}(g)$. Hence, if $G(S)$ is a quasi-clique in G and $S\supset V(g)$, $S\subseteq V(g)\cup V^G_{vad}(g)$. Accordingly,we only need to use the vertices in $V^G_{vad}(g)$ to grow g, which can further improve the algorithm's efficiency.

Assume g is a subgraph of a graph G, for a vertex $u{\in}\ V(g)$, let $V^g_{ext}(u){=}N^G(u)\cap V^G_{vad}(g)$, and the **extensible degree** be the cardinality of $V^g_{ext}(u)$, i.e., $extdeg^g(u){=}|V^g_{ext}(u)|$. If there exists a critical vertex v in g such that $extdeg^g(v)=0$, we call v a failed-vertex of g.

LEMMA 3.7. **(Critical Connectivity Pruning)** *If there exists a failed-vertex v in a k-subgraph g of G, there will be no such an induced subgraph Q of G that $V(Q){\supset}V(g)$ and Q is a quasi-clique.*

PROOF. We prove this by contradiction. Let Q be such an induced subgraph of G and $|Q|=m$. Obviously $m>k$ and $V(Q)\subseteq V(g){\cup}V^G_{vad}(g)$. Since v is a critical vertex in g, $deg^g(v)=\lceil\gamma\cdot(k-1)\rceil$ and $\lceil\gamma\cdot(k-1)\rceil<\lceil\gamma\cdot k\rceil$. Furthermore, because $deg^Q(v)\leq deg^g(v)+extdeg^g(v)$ and $extdeg^g(v)=0$, $deg^Q(v)<\lceil\gamma\cdot k\rceil\leq\lceil\gamma\cdot(m-1)\rceil$, which contradicts with the assumption. Thus, there must exist no such an induced subgraph. □

Given a k-subgraph g of a graph G, if $\exists u \in V(g)$ such that $deg^g(u) < \lceil \gamma \cdot (k-1) \rceil$ and $deg^g(u) + extdeg^g(u) < \lceil \gamma \cdot (k - 1 + extdeg^g(u)) \rceil$, we call u an unpromising vertex in g.

LEMMA 3.8. **(Subgraph Connectivity Pruning)** *If a k-subgraph g of G contains an unpromising vertex u, there will be no induced subgraph Q of G such that $V(Q) \supset V(g)$ and Q is a quasi-clique.*

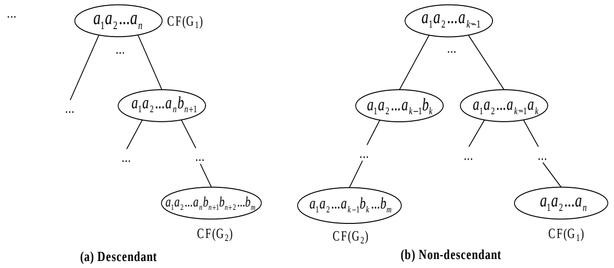
PROOF. Let Q be such an induced subgraph and $|Q| = l$. Since Q is a γ-quasi-clique, $V(Q) \subseteq V(g) \cup V^G_{vad}(g)$ holds. Let $V' = V(Q) \cap V^g_{ext}(u)$ and $|V'| = m$, then $l \geq m+|V(g)|=m+k$. Because $deg^g(u) < \lceil \gamma \cdot (k-1) \rceil$ and $deg^g(u) + extdeg^g(u) < \lceil \gamma \cdot (k - 1 + exdeg^g(u)) \rceil$, from Lemma 3.3 we can get that $\forall i \in [0, exdeg^g(u)], deg^g(u) + i < \lceil \gamma \cdot (k-1+i) \rceil$. Thus, $deg^Q(u) = deg^g(u) + |V'| = deg^g(u) + m < \lceil \gamma \cdot (k-1+m) \rceil \leq \lceil \gamma \cdot (l-1) \rceil$, that is, $deg^Q(u) < \lceil \gamma \cdot (|Q|-1) \rceil$. This contradicts with the assumption that Q is a γ-quasi-clique. □

According to Lemma 3.7 and Lemma 3.8, if a subgraph contains a failed-vertex or an unpromising vertex, it does not make any contribution to the generation of quasi-cliques. Thereby, after getting the valid extensible candidate set $V^G_{vad}(g)$, once we inspect the existence of failed-vertices or unpromising vertices in g, then there is no hope to grow instance g to generate quasi-cliques, and thus we can stop growing g.

3.4 Closure Checking Scheme

By integrating the pruning techniques proposed in this paper with the vice-frequent subgraph enumeration framework, we can discover the complete set of frequent quasi-cliques. How do we discover the closed quasi-cliques? A straightforward approach is to store all the frequent quasi-cliques that we have found, and when inserting a quasi-clique q to the result, we do *super-clique-detecting* which checks whether there already exists an element q' such that can subsume the current q and *sub-clique-detecting* which checks if there exists any already mined quasi-clique q' that can be subsumed by q. Obviously, this naïve approach is very costly, we will introduce a more efficient closure checking scheme.

Given two subgraphs, G_1 and G_2, with canonical forms $CF(G_1) = a_1a_2...a_n$ and $CF(G_2){=}b_1b_2...b_m$ (where $n{<}m$) respectively, if $CF(G_1){\sqsubset}CF(G_2)$, there must exist n integers $1{\leq}i_1{<}i_2{<}...{<}i_n{\leq}m$ such that $a_1{=}b_{i_1}$, $a_2{=}b_{i_2}$, ..., $a_n{=}b_{i_n}$. If $i_n{=}n$, the relationship of $CF(G_1)$ and $CF(G_2)$ in the enumeration tree can be illustrated in Figure 3(a), that is, the node corresponding to $CF(G_1)$ is an ancestor of the node corresponding to $CF(G_2)$. Otherwise, let $k = min\{j \mid i_j{\neq}j\}$, then $CF(G_1)$ and $CF(G_2)$ have the same prefix $a_1a_2...a_{k-1}$(if $k{=}1$, the prefix is empty) and $a_k{>}b_k$ (as $a_k{=}b_{i_k}$,$i_k{>}k$, and $b_k{<}b_{i_k}$ hold), thus $CF(G_1){>}CF(G_2)$ holds, and this relationship in the enumeration tree is shown in Figure 3(b).



(a) Descendant

(b) Non-descendant

Figure 3: Two Cases of $CF(G_1) \sqsubset CF(G_2)$ in Vice-Frequent Subgraph Enumeration Tree

One strategy to speed up the pattern closure checking is that we postpone the closure checking for G_1 until all its descendants have been processed. In this way, G_2 must be discovered before G_1 for the first case shown in Figure 3(a), as the node $CF(G_2)$ is a descendant of $CF(G_1)$. In the second case as shown in Figure 3(b), it is evident that G_2 is also discovered before G_1 according to the DFS traverse strategy. In summary, if the current quasi-clique G_1 can be subsumed by another frequent closed γ-quasi-clique G_2 (i.e., $CF(G_1) \sqsubset CF(G_2)$ and $sup^D_s(G_1) \leq sup^D_s(G_2)$), the insertion of G_2 into the result set will occur before the closure checking of G_1. Accordingly, when we check if a frequent quasi-clique q is closed or not, there is no need to perform *sub-clique-detecting*, as there does not exist any quasi-clique q' in the result set such that $CF(q') \sqsubset CF(q)$ and $sup^D_s(q') \leq sup^D_s(q)$.

Although there is no need for *sub-clique-detecting*, we still have to perform *super-clique-detecting*. As shown in Figure 3, there are two cases for *super-clique-detecting*. In the first case, we need to check if there exists a descendant quasi-clique G_2 of the current quasi-clique G_1 that can subsume G_1 in the enumeration tree. According to our strategy described above, we know G_1 must be mined after all its descendants, the *super-clique-detecting* in this case becomes relatively simple. After the processing of all descendants of G_1, we let the recursive mining procedure return the maximum strict-support (denoted by r) of all frequent quasi-clique nodes under the subtree rooted with $CF(G_1)$ (if there does not exist any frequent quasi-clique, then it returns value zero). If $sup^D_s(G_1) \leq r$, we know G_1 is non-closed and will not insert G_1 into the result set. However, if $sup^D_s(G_1){>}r$, we still need to check if there exists any non-descendant super-clique of G_1 that can subsume G_1 (i.e., the second case shown in Figure 3(b)).

In order to accelerate the non-descendant super-clique-detecting process, we divide the elements in the result set into different groups according to their absolute strict-support. In each group, we first order them by the size of the quasi-cliques in descending order, and among the quasi-cliques with the same size in the same group, we then order them by their canonical form in descending order. This processing can accelerate the comparison steps.

3.5 The Algorithm

In the following we describe the Cocain algorithm by integrating various techniques discussed earlier. Let us first introduce the SUB-ALGORITHM 1:**Valid**, which is called by Cocain in order to compute the valid extensible candidates for an instance subgraph. For each vertex u in current instance g, we scan the graph G in which g resides to find the set $D(u)$ (line 06), and refine $D(u)$ based on the combination pruning technique (line 07). Then we generate the set $I(u)$ from $D(u)$, obtain the extensible candidate set T of u (lines 08-10), and conjoin each discovered extensible candidate set to get the global extensible candidates rs (line 11). Finally, we apply the vertex connectivity pruning (line 12), critical connectivity pruning, and subgraph connectivity pruning (lines 13-14) to rs to generate the final set of valid extensible candidates w.r.t. g.

Before running **Cocain** as shown in ALGORITHM 1, we first compute the set of vice-frequent vertex labels and their corresponding instances by scanning the original database, and remove from the graph database the vertices with non-vice-frequent vertex labels. This procedure can reduce the size of input graphs significantly, especially when *min_sup* is high. After this preprocessing, we use Cocain to mine the complete set of frequent closed coherent quasi-cliques. For the current prefix vice-frequent subgraph g, we first use procedure **Valid** to get the set of valid extensible candidates V_{vad} for each instance of g (lines 17-18), from which we can further calculate the vice frequent extensible labels (line 19). For each vice-frequent extensible label, we invoke Cocain to discover descendants of g (lines 21-23). After all recursive invocations have returned, we can use the closure checking scheme to determine whether or not to insert g to the final result set according to the strict-support of g and the returned values of the recursive invocations (lines 24-25).

SUBALGORITHM 1: **Valid**(g)

```
INPUT:
    (1) g: an instance subgraph.
OUTPUT:
    (1) rs: the set of valid extensible candidates w.r.t. g.
BEGIN
01. set $rs = V(G)$(G is the graph in which g resides);
02. For each vertex $u$ in $V(g)$
03.     If rs is empty
04.         break;
05.     set $T = D = I = \phi$;
06.     $D = \{v|dis^G(u,v) = 1\}$;
07.     Refine $D$ using combination pruning;
08.     $I = \{v \mid \exists\, t \in D,\ dis^G(t,v) = 1$ and $v \notin D\}$;
09.     $T = D \cup I$;
10.     Remove each element $v' \in T$ which satisfies $L(v') < LAS(g)$;
11.     $rs = rs \cap T$;
12. Remove invalid extensible candidates from $rs$;
13. If there exists a failed or an unpromising vertex in $g$
14.     $rs = \phi$;
15. return rs;
END
```

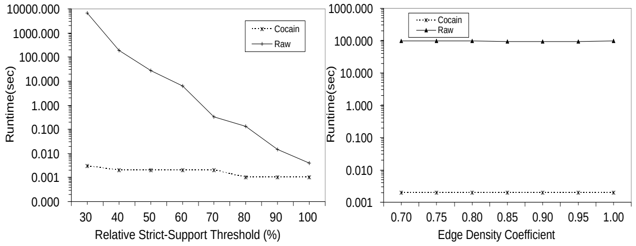
ALGORITHM 1: **Cocain**(D, $CF(g)$,$INS(g)$,min_sup,γ)

```
INPUT:
    (1) D: the input graph database,
    (2) $CF(g)$: the canonical form of g,
    (3) $INS(g)$: the set of instances of g in the database D,
    (4) $min\_sup$: the minimum support threshold,
    (5) $\gamma$: the edge density coefficient.
OUTPUT:
    (1) rs: the set of frequent closed $\gamma$-quasicliques,
    (2) max: the maximum strict-support of all descendant quasi-cliques of $g$.
BEGIN
16. glbsup=0, rv=0;
17. For each instance $ins \in INS(g)$
18.     $V_{vad}(ins)$ = Valid($ins$);
19. Calculate vice-frequent valid candidate label set $VEX(g)$ according to each
        $V_{vad}(ins)$;
20. Sort the labels in $VEX(g)$ in a certain order;
21. For each label $l \in VEX(g)$
22.     $rv = Cocain(D, CF(g) \diamond l, INS(g \diamond l), min\_sup, \gamma)$;
23.     $glbsup = max\{glbsup, rv\}$;
24. If $(sup_s(g) \geq min\_sup)$ and $(sup_s(g) > glbsup)$
25.     Insert $CF(g)$ into $RS$ if g passes the non-descendant super-clique-detecting;
26. return $max\{sup_s(g), glbsup\}$;
END
```

4. EMPIRICAL RESULTS

We conducted an extensive performance study to evaluate various aspects of the algorithm. We implemented the algorithm in C++, and all experiments were performed on a PC running FC 4 Linux and with 1.8GHz AMD Sempron CPU and 1GB memory. In the experiments, we used the US stock market series database [2], which was converted to a set of graphs based on the same method of [5]. Due to limited space, here we only report the results w.r.t. a correlation coefficient of 0.99.

Efficiency Test. We implemented one baseline algorithm, Raw, which excludes three pruning techniques, *combination pruning, critical connectivity pruning*, and *subgraph connectivity pruning*. By comparing the runtime efficiency between Cocain and Raw, we can get an idea about the effectiveness of the pruning techniques proposed in this paper. Figure 4 shows the runtime comparison between Cocain and Raw by fixing γ at 1.0 and varying *min_sup*, and fixing min_sup at 40% and varying γ respectively. We see that Cocain is always faster than Raw. The high performance of Cocain in comparison with Raw also demonstrates that the pruning techniques proposed for Cocain are extremely effective.



a) $\gamma = 1.0$ b) min_sup=40%

Figure 4: Efficiency Comparison (stock market data)

Scalability Test. Meanwhile, a comprehensive scalability study was conducted in terms of the base size. The results show that Cocain has linear scalability in runtime against the number of input graphs in database.

5. CONCLUSION

In this paper we proposed a novel algorithm, Cocain, to mine frequent closed coherent quasi-cliques from large dense graph databases. By focusing on vertex labels, we first introduced a simple canonical form to uniquely represent a quasi-clique pattern. By exploring some nice properties of quasi-clique patterns, we proposed several effective search space pruning techniques, *diameter pruning, combination pruning, vertex connectivity pruning, critical connectivity pruning, and subgraph connectivity pruning*, which could be used to accelerate the mining process. We also introduced an efficient pattern closure checking scheme to speed up the discovery of closed quasi-cliques. An extensive performance study with both real and synthetic databases has demonstrated that Cocain is very efficient and scalable.

6. ACKNOWLEDGEMENT

The authors are grateful to Vladimir L. Boginski, Panos M. Pardalos, and Sergiy Butenko for providing us the US stock market database. We thank Jasmine Zhou and Haiyan Hu for sending us the yeast microarray database and Michihiro Kuramochi for sending us the KEGG database. Thanks also go to Beng Chin Ooi, Anthony K. H. Tung and Xifeng Yan for their kind help.

7. REFERENCES

[1] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *VLDB'94*.

[2] V. Boginski, S. Butenko, P.M. Pardalos. On structural properties of the market graph. In *A. Nagurney (editor), Innovations in Financial and Economic Networks*, Edward Elgar Publishers, Apr. 2004.

[3] H. Hu, X. Yan, Y. Hang, J. Han, X. Zhou. Mining coherent dense subgraphs across massive biological network for functional discovery. *Bioinformatics, Vol. 21, Suppl. 1, 2005*.

[4] J. Pei, D. Jiang, A. Zhang. On mining cross-graph quasi-cliques. *SIGKDD'05*.

[5] J. Wang, Z. Zeng, L. Zhou. CLAN:An Algorithm for Mining Closed Cliques From Large Dense Graph Databases. *ICDE'06*.

[6] X. Yan, X. Zhou, J. Han. Mining closed relational graphs with connectivity constraints. *SIGKDD'05*.