BIOINFORMATICS

Vol. 1 no. 1 2005
Pages 1–1

# Mining coherent dense subgraphs across massive biological networks for functional discovery

Haiyan Hu[1], Xifeng Yan[2], Yu Huang[1], Jiawei Han[2], and Xianghong Jasmine Zhou[1*]

[1]Program in Molecular and Computational Biology, University of Southern California, Los Angeles, CA 90089, USA and [2]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801

**ABSTRACT**

**Motivation:** The rapid accumulation of biological network data translates into an urgent need for computational methods for graph pattern mining. One important problem is to identify recurrent patterns across multiple networks to discover biological modules. However, existing algorithms for frequent pattern mining become very costly in time and space as the pattern sizes and network numbers increase. Currently, no efficient algorithm is available for mining recurrent patterns across large collections of genome-wide networks.

**Results:** We developed a novel algorithm, CODENSE, to efficiently mine frequent coherent dense subgraphs across large numbers of massive graphs. Compared to previous methods, our approach is scalable in the number and size of the input graphs, and adjustable in terms of exact or approximate pattern mining. Applying CODENSE to 39 co-expression networks derived from microarray data sets, we discovered a large number of functionally homogenous clusters and made functional predictions for 169 uncharacterized yeast genes.

**Availability:** http://zhoulab.usc.edu/CODENSE/

**Contact:** xjzhou@usc.edu

## 1 INTRODUCTION

The recent development of high-throughput technologies provides a range of opportunities to systematically characterize diverse types of biological networks. "Network Biology" has been an emerging field in biology. The variety of biological networks can be classified into two categories: (1) *physical networks*, which represent physical interactions among molecules, e.g., protein-interaction, protein-DNA interaction, and metabolic reactions; and (2) *conceptual networks*, which represent functional associations of molecules derived from genomic data, e.g., co-expression relationships extracted from microarray data, and genetic interactions obtained from synthetic lethality experiments. While the physical network data is as-yet very limited in size, the large amount of microarray data allows us to infer conceptual functional associations of genes under various conditions for many model organisms, thus providing valuable information to study the functions and the dynamics of biological systems.

Studying the building principles of biological networks could potentially revolutionize our view of biology and disease pathologies (Barabasi & Oltvai, 2004). The popular clustering approach can draw densely connected modules from biological networks, which are often biologically meaningful, e.g., a dense protein interaction subnetwork may correspond to a protein complex (Bader & Hogue, 2003; Spirin & Mirny, 2003), and a dense co-expression network may represent a tight co-expression cluster (Sharan & Shamir, 2000). Due to the noisy nature of high-throughput data, a significant number of spurious edges exist in biological networks, which may lead to the discovery of false patterns. Since biological modules are expected to be active across multiple conditions, we can easily filter out spurious edges by mining frequent patterns in multiple biological networks simultaneously. A straightforward approach is to aggregate these networks together and identify dense subgraphs in the aggregated graph. However, it could result in false dense subgraphs that may not occur frequently in the original networks. Figure 1a illustrates such an example with a cartoon of six graphs. If we simply add these graphs together to construct a summary graph, we may find a dense subgraph comprising vertices *a, b, c,* and *d*. Unfortunately, this subgraph is neither dense nor frequent in the original graphs.

A potential solution to the false pattern problem is mining frequent subgraphs directly. A subgraph is frequent if it occurs multiple times in a set of graphs. Frequent subgraph discovery in general is considered a hard problem. However, biological networks can often be modeled as a special class of graph where each gene occurs once and only once in a graph. That means, our graph has distinct node labels, and we do not have the "subgraph isomorphism problem" which is NP-hard and so far constitutes

* To whom correspondence should be addressed.

the bottleneck of subgraph frequency counting. We term such graphs *relation graphs*. Recently, we and others have designed efficient approaches to identify frequent subgraphs across multiple relation networks by decomposing the networks into smaller pieces and applying pattern expansion techniques (Kuramochi & Karypis, 2004; Yan *et al.*, 2005), or by performing frequent set mining with subsequent connectivity checking (Koyuturk *et al.*, 2004). However, these approaches encounter scalability and interpretability issues when being applied to massive biological networks: (1) In both approaches we tested, the time and memory requirements increase exponentially with increasing size of patterns and increasing number of networks. The number of frequent dense subgraphs is explosive when there are very large frequent dense subgraphs, e.g., subgraphs with hundreds of edges. (2) A frequent dense subgraph may not represent a tight association among its nodes. Figure 2 shows a sample network data set. Vertices *e, c, f, h, d,* and *g* form a frequent dense subgraph. However, biologically it is more interesting to divide this subgraph into two modules, one comprising *e, c, f,* and *h*; the other comprising *h, d, g,* and *e* since these two modules have different occurrences throughout this graph set (details see the figure caption). As one can see, frequent dense subgraphs may not capture accurate information for the discovery of biological modules.

In this paper, we address the aforementioned two issues and develop a novel algorithm, called "*CODENSE*", to mine **co**herent **dense** subgraphs, a concept having better interpretability than frequent graph. All edges in a coherent subgraph should exhibit correlated occurrences in the whole graph set. We also term this kind of subgraph "*network module*". According to the definition of coherent dense subgraph, we are able to distinguish the two modules shown in Figure 2. Moreover, the design of CODENSE can solve the scalability issue. Instead of mining each biological network individually, CODENSE compresses the networks into two meta-graphs and performs clustering in these two graphs only. Thus, CODENSE can handle any large number of networks. Using CODENSE, we can successfully identify high-quality network modules within limited time and memory.

As a side product, CODENSE also provides a solution to a graph mining problem—discovery of overlapping graph clusters. It is known that under different conditions, one gene may serve different roles and be involved in different functional groups (Gasch & Eisen, 2002); thus identifying overlapping clusters is important in biological applications. However, most graph clustering algorithms follow the methodology of graph partitioning (Spirin & Mirny, 2003; Van Dongen, 2000); they usually cannot identify overlapping clusters. For example, in Figure 1b, two cliques {a, b, c, d, e, f} and {e, f, h, i} share two common vertices {e, f}. If a partition-based method first identified the former clique, the latter clique will be missed. Here, as a component of CODENSE, we designed

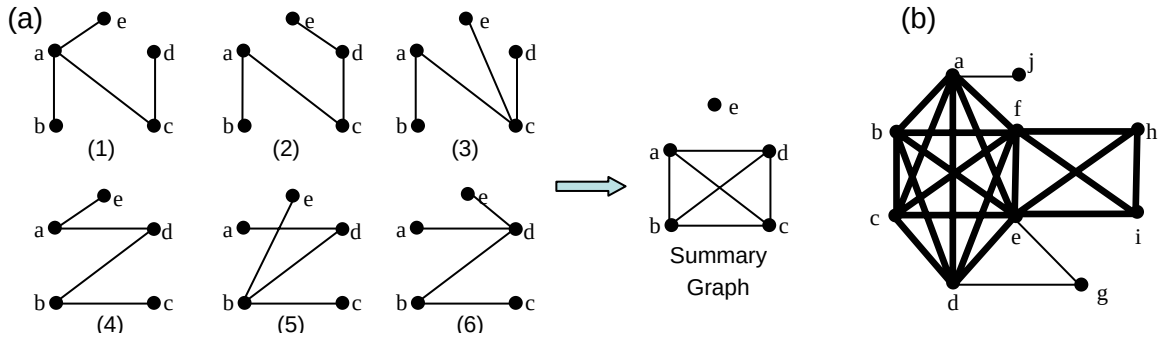


**Figure 1.** Given six graphs with the same vertex set but different edge sets, we construct a summary graph by adding these six graphs together and by deleting edges that occur less than three times in the graphs. The dense subgraph in the summary graph *{a, b, c, d}* actually does not occur in any original graph. (b) The vertices *e* and *f* are shared by cliques *{a, b, c, d, e, f}* and *{e, f, h, i}*; they can be assigned to both cliques only by approaches that are able to detect overlapping dense subgraphs (cliques are the densest subgraphs).

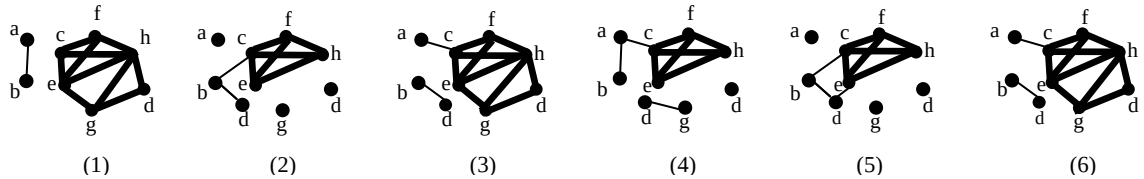


**Figure 2.** Shown are six graphs with the same vertex set but different edge sets. The bold subgraph *{c-e, c-f, c-h, f-e, f-h, e-h, e-g, g-h, h-d, g-d}* occurs in three out of the six graphs (graphs *1,3,* and *6*). However, the vertices/edges of this subgraph may not be tightly associated in their occurrence, because one large component, the subgraph *{c-e, c-f, c-h, f-e, f-h, e-h}* occurs in every network..

a novel algorithm, MODES (Mining Overlapping DEnse Subgraphs), to identify overlapping graph clusters.

As an application example, we used CODENSE to identify frequent co-expression clusters across multiple microarray data sets. A microarray data set is modeled as an unweighted and undirected network, where each gene is represented by one node and two genes are connected with an edge if they show high expression correlation. A densely connected subgraph in these networks corresponds to a tight co-expression cluster. However, several studies pointed out that clusters derived from a *single* microarray data set often include spurious links and may not be functionally homogenous (Allocco *et al.*, 2004; Clare & King, 2002) A recent study showed that genes co-expressed in multiple datasets tend to have the same functions (Lee *et al.*, 2004). Here, applying our methods to 39 microarray data sets of *S. cerevisiae*, we demonstrated that recurrent expression clusters are very likely to be homogenous in function and regulation, and can be used to perform large-scale functional annotation of uncharacterized genes.

The remainder of this paper is organized as follows. Section 2 gives the problem formulation. Our algorithms of mining coherent dense subgraphs and overlapping dense subgraphs are examined in Section 3 and Section 4, respectively. We give a thorough comparison between our approach and the others in Section 5. The experimental study and biological applications are examined in Section 6. Section 7 concludes our study.

## 2 PROBLEM FORMULATION

A relation graph set consists of $n$ undirected simple graphs, $D=\{G_i = (V,E_i)\}$, $i = 1,\ldots,n$, $E_i \subseteq V\times V$, where a common vertex set $V$ is shared by the graphs in the set. We denote the vertex set of a graph $G$ by $V(G)$ and the edge set by $E(G)$. Let $w_i(u, v)$ be the weight of an edge $e_i(u, v)$ in $G_i$. For an unweighted graph, $w_i(u, v)$ is equal to $1$ if there is an edge between $u$ and $v$, otherwise $0$. We choose to illustrate the principles on unweighted and undirected graphs in this paper, although our algorithm should be extendable to weighted and directed graphs.

**Definition 1 (Support)** Given a relation graph dataset, $D=\{G_1,G_2,\ldots,G_n\}$, where $G_i=(V,E_i)$, the support of a graph $g$ is the number of graphs (in $D$) where $g$ is a subgraph, written *support(g)*. A graph is frequent if its support is greater than a minimum support threshold.

**Definition 2 (Summary Graph)** Given a relation graph dataset, $D=\{G_1,G_2,\ldots,G_n\}$, where $G_i=(V, E_i)$, the summary graph of $D$ is an unweighted graph $\hat{G}=(V, \hat{E})$ where an edge is present if it occurs in more than $k$ graphs in $D$, where $k$ is a user-defined support threshold (see an example in Figure 1a).

**Definition 3 (Edge Support Vector)** Given a relation graph dataset, $D=\{G_1,G_2,\ldots,G_n\}$, where $G_i=(V, E_i)$, the support vector of an edge $e$, written $\mathbf{w}(e)$, is of length $n$ where $n$ is the number of graphs. The $i$-th element of $\mathbf{w}(e)$ corresponds to the weight of edge $e$ in the $i$-th graph.

The support vector of the edge *(a, b)* for the six graphs shown in Figure 1a is *[1, 1, 1, 0, 0, 0]*, while the support vector of the edge *(b, c)* is *[0, 0, 0, 1, 1, 1]*. As one can see, edges *(a, b)* and *(b, c)* are not correlated in this dataset, though both of them are frequent.

We use a special graph, termed *second-order graph* (denoted as $S$), to illustrate the co-occurrence of edges across all graphs in a relation graph set $D$. Each edge in $D$ is transformed into a vertex in $S$, and two vertices $u$ and $v$ in $S$ will be connected if their corresponding edge support vectors $\mathbf{w}(u)$ and $\mathbf{w}(v)$ in $D$ show high similarity. Depending on whether or not the edges are weighted, the similarity measure could be Euclidean distance or Pearson's correlation. Figure 3 (Step 3b) shows how to generate a second-order graph from a set of edge support vectors. For example, the Euclidean distance between the support vectors of edges *(c, e)* and *(c, i)* is only 1, so we create an edge between the vertices *(c, e)* and *(c, i)* in the second-order graph $S$ shown in Figure 3. In contrast to the second-order graph, we term the original graphs $G_i$ the *first-order graphs*. The utilization of second-order graph discussed in this paper is one type of second-order analysis, a concept that has been proposed in our previous publication (Zhou *et al.*, 2005).

**Definition 4 (Second-Order Graph)** Given a relation graph dataset, $D=\{G_1,G_2,\ldots,G_n\}$, where $G_i = (V,E_i)$, the second-order graph is an unweighted graph $S=(V\times V,E_s)$, where the vertex set of $S$ is the edge set of $G$, and an edge connects vertices $u$ and $v$ if the similarity between the corresponding edge support vectors $\mathbf{w}(u)$ and $\mathbf{w}(v)$ is greater than a threshold.

In reality, if $G_i$ is large and dense, $S$ will be impractically large. Therefore, to achieve efficiency, in this paper we construct $S$ each time only for a subgraph of the summary graph $\hat{G}$.

**Definition 5 (Coherent Graph)** Given a relation graph dataset, $D=\{G_1,G_2,\ldots,G_n\}$, where $G_i=(V,E_i)$, a subgraph $sub(\hat{G})$ is coherent if all the edges of $sub(\hat{G})$ have support higher than $k$ and the second-order graph of $sub(\hat{G})$ is dense.

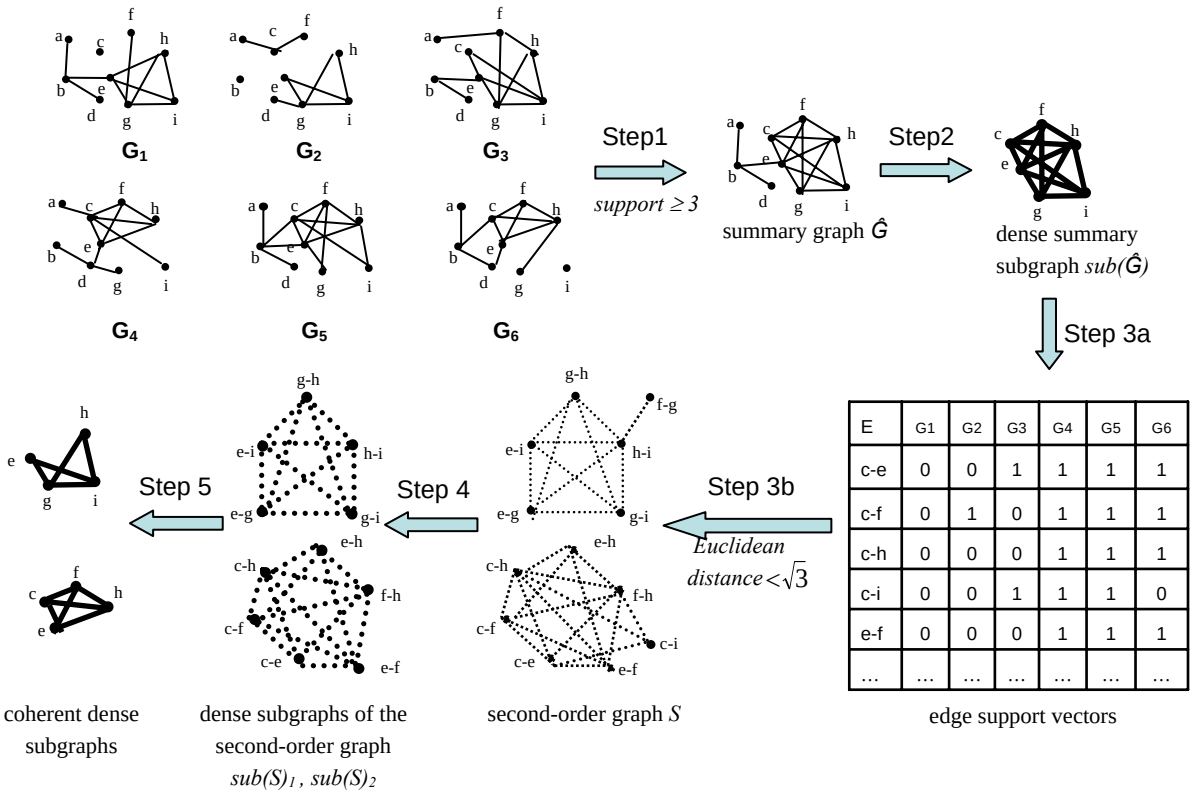


**Figure 3. CODENSE:** Discover Coherent Dense Subgraphs Across Multiple Graphs (dense subgraphs are marked bold).

**Definition 6 (Graph Density)** The density of a graph *g*, written *density(g), is 2m/(n(n-1))* where *m* is the number of edges and *n* the number of vertices in *g*.

The problem of mining coherent dense subgraphs is formulated as follows: *given a relation graph dataset, $D=\{G_1,G_2,\ldots,G_n\}$, discover subgraphs g that satisfy the following two criteria simultaneously: (1) g is a densely connected subgraph of the summary graph; and (2) g is a coherent graph.* As discussed in the Introduction section, the coherent dense subgraphs have significant biological interests.

## 3 CODENSE: MINING COHERENT DENSE SUBGRAPHS

A scalable algorithm for mining coherent dense subgraphs must perform well despite increasing number of graphs and increasing size of patterns. In order to tackle this problem, we investigate the relation between a coherent dense subgraph and the summary graph as well as its second-order graph. We make the following two observations:

**(1)** If a frequent subgraph is dense, then it must be a dense subgraph in the summary graph. However, the reverse conclusion is not true. A dense subgraph in the summary graph may be neither frequent nor dense in the original data set (see Figure 1a for an example).

**(2)** If a subgraph is coherent (its edges show high correlation in their occurrences across a graph set), then its $2^{nd}$-order graph must be dense.

These two observations provide a clue to mining coherent dense subgraphs with reasonable computational cost. According to Observation 1, each frequent dense subgraph is a subgraph of a dense summary graph. We can start from the summary graph and mine dense summary subgraphs first. Once it is done, we can single out coherent subgraphs from dense summary subgraphs by mining their corresponding second-order graphs.

Our CODENSE algorithm consists of five steps, as outlined in Algorithm 1 and illustrated in Figure 3. In Steps 2, 4, and 5, a novel algorithm to discover overlapping dense subgraphs, called "MODES," is employed. We will describe the design of the MODES algorithm in Section 4.

In Step 1, CODENSE builds a summary graph by eliminating infrequent edges.

In Step 2, CODENSE identifies dense subgraphs (possibly overlapping) in the summary graph. While the dense subgraphs in the summary graph may not be truly frequently occurring in the original graphs, they do serve as a superset of potential frequent dense subgraphs in the original graphs. We start with this superset to refine the search results.

**ALGORITHM 1: CODENSE**

1: build a summary graph $\hat{G}$ across multiple relation graphs $G_1, G_2, \ldots, G_n$;
2: mine dense summary subgraphs *sub($\hat{G}$)* in $\hat{G}$ using MODES;
**for** each dense summary subgraph *sub($\hat{G}$)* **do**
3: •construct the second-order graph *S*;
4: •mine dense subgraphs *sub(S)* in *S* using MODES;
5: •**for** each dense subgraph *sub(S)* **do**
*convert *sub(S)* into the first-order graph *G*;
*mine dense subgraphs *sub(G)* in *G* using MODES;
*output *sub(G)*;

In Step 3, CODENSE builds a second-order graph for each dense summary subgraph.

In Step 4, CODENSE identifies dense subgraphs in the second-order graph. The high connectivity among vertices in the second-order graph indicates that the corresponding edges show high similarity in their occurrences across the *n* original graphs.

In Step 5, CODENSE discovers the real coherent dense subgraphs. Although a dense subgraph *sub(S)* found in Step 4 guarantees the co-occurrence of its edges across the *n* relation graphs, those edges may no longer form a densely connected graph in the original summary graph. To eliminate such cases, we convert the vertices in *sub(S)* back to edges and then apply the MODES algorithm to identify dense subgraphs. The resulting subgraphs by construction will satisfy the criteria for coherent dense subgraphs: (1) they are dense subgraphs and all of their edges occur frequently; and (2) their edges are highly correlated in their occurrences across the *n* relation graphs.

In comparison with previous frequent graph mining algorithms, our algorithm may not show a significant advantage when the number of relation graphs *n* is small. However, when *n* is large, CODENSE will achieve significant time and memory efficiency since it works on two graphs only: summary graph and second-order graph instead of the *n* graphs. On the contrary, traditional frequent pattern mining algorithms will not work well for high numbers of large graphs due to the astronomical number of frequent patterns.

In case the relation graphs do not contain a large amount of edges, we can skip the step of clustering the summary graph, and start from Step 3 by transforming all edges directly into the second-order graph with further steps unchanged. The clustering of the summary graph serves the purpose of restricting the second-order graph to a reasonable size to avoid excessive computation.

## 4 MODES: MINING OVERLAPPING DENSE SUBGRAPHS

The overlapping dense subgraph mining algorithm, called MODES, is frequently used in CODENSE. In this section, we present the details of its design. MODES is developed based on HCS (Mining Highly Connected Subgraphs) (Hartuv & Shamir, 2000), with two new features: (1) MODES is more efficient in identifying dense subgraphs; and more importantly, (2) MODES can discover overlapping subgraphs.

It is computationally intractable to enumerate all dense subgraphs in a large graph. Usually, a large graph is first clustered hierarchically and those dense clusters are singled out. HCS is this kind of algorithm. It recursively partitions the graph into two subgraphs until its minimum cut is no less than half of its vertex set size.

**Definition 7 (Minimum Cut)** Given a graph *G*, an edge cut is a set of edges $E_c$ such that $E(G)-E_c$ is disconnected. A minimum cut is the smallest set in all edge cuts.

There are two issues remaining in HCS. First, HCS can not identify overlapping dense subgraphs because of its nature of graph-partitioning. Second, the above recursive partitioning process is time-consuming. The fastest deterministic minimum cut algorithm in practice has time complexity $O(|V||E| + |V|^2 log|V|)$, where $|V|$ and $|E|$ are the vertex set size and the edge set size of a given graph (Stoer & Wagner, 1997). The minimum cut criterion adopted by HCS favors cutting small sets of nodes from the graph. When applying the algorithm repeatedly to a large graph consisting of more than thousands of vertices, such unbalanced cuts could lead to unexpected high costs. As verified by our experiments, we found HCS often cuts off one node in each iteration, thus having time complexity $O(|V|^2|E|+ |V|^3 log|V|)$.

To avoid the undesirable bias for partitioning out small sets of vertices and to speed up the process, we apply the normalized cut (Shi & Malik, 2000) instead of minimum cut in the initial runs of the HCS algorithm. Normalized cut is able to better balance the sizes of the partitions. When the size of the partitions generated by normalized cut is reasonably small, we proceed with the minimum cut algorithm to identify dense subgraphs. We revert to the use of the minimum cut in the later stage to better exploit its power in clustering without severe effects on computational cost.

In order to identify overlapping dense subgraphs, we designed the following procedure, as outlined in Algorithm 2 and illustrated in Figure 4: (1) We mine dense subgraphs using the above modified HCS algorithm in a given graph *G*. (2) Each discovered dense subgraph *sub(G)* is then condensed into a single vertex *v*'; any vertex *v* that does not belong to *sub(G)* will have an edge with *v*' if *v* has

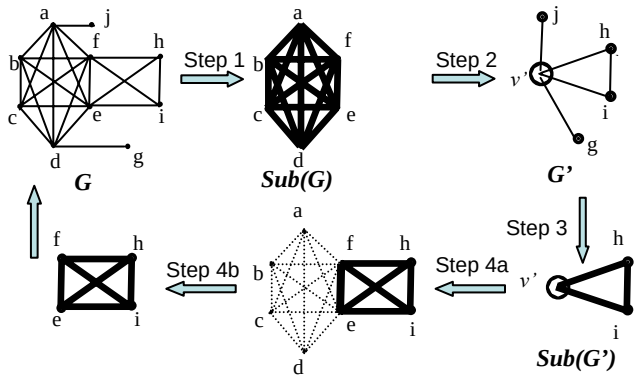


**Figure 4. MODES**: Mine Overlapping Dense Subgraphs.

edge with a vertex in *sub(G).* (3) The condensed graph, written *G',* is then re-clustered using the above modified HCS algorithm. (4) Once the clustering is done, if any newly discovered dense subgraph *sub(G')* contains condensed vertices, MODES restores the condensed vertices back into subgraphs. To avoid the repetitive discovery of already discovered dense subgraphs, MODES conducts the following to focus on the vertices that have not been clustered previously: for a restored subgraph *C*, MODES removes the vertices $v$ ($v \in C$) where $v$ is connected to less than $p\%$ of the vertices in $V(sub(G'))-V(C)$. Since the resulting subgraph *sub(G')* may not be dense any more, in order to extract the dense subgraphs, we perform Step 1 again on *sub(G')* to identify its dense subgraphs. This procedure is repeated until no new dense graph is discovered. To speed up the computation, we require that a new dense subgraph should have less than *90%* overlap with any existing dense subgraph.

The running time of computing normalized cut is determined by the extraction of the second smallest eigenvalue of $I-D^{-1/2}WD^{-1/2}$, where W(i, j) is the weight of edge (i, j), D(i, i) = $\sum_j$ W(i, j). Its complexity is $O(n^c)$, $1 \le c \le 3$, ($O(n^3)$ is the upper bound on solving the eigensystem of an $n$-dimension square matrix) (Shi & Malik, 2000). Empirically, the value of $c$ is around *1.5* when the graph is very sparse (Shi & Malik, 2000). Assume a graph $G = (V, E)$ is clustered to $k$ partitions $V_1, V_2, \ldots V_k$, where $V_1 \cup V_2 \cup \ldots \cup V_k = V$. The total cost of clustering these $k$ partitions is $O(|V_1|^c+|V_2|^c + \ldots + |V_k|^c)$ which is at most $O(|V|^c)$ given $c \ge 1$. For hierarchical recursive clustering, the cost is $O(d|V|^c)$, where $d$ is the largest recursive steps. The value of $d$ is between $O(log|V|)$ and $O(|V|)$. Since normalized cut prefers balanced partitions, in general, $d$ is far away from $|V|$ but close to $log\ |V|$, which means our algorithm MODES can run much faster than the worst case.

MODES may iterate several times in order to discover overlapping dense subgraphs. Each iteration involves a hierarchical clustering. Assume that the maximum number of recursions for a given graph is $r$ (Algorithm 2, MODES). The running time of MODES is $O(r|V|^{2.5})$ for a sparse graph. In practice, we may restrict the recursion depth.

**ALGORITHM 2: MODES**

1: mine dense subgraphs, *sub(G)*, in a given graph *G* using the modified HCS algorithm in which normalized-cut and min-cut are combined for recursive partitions;
2: **for** each dense subgraph *sub(G)* **do**
- condense *sub(G)* into a condensed vertex v' in the original graph;

denote the condensed graph by *G';*
3: mine dense subgraphs *sub(G')* from the condensed graph *G'* using the modified HCS algorithm again;
4: **for** each dense subgraph *sub(G')* **do**
**if** *sub(G')* contains any condensed vertex **then**
(a) restore each condensed component *C* in *sub(G')* ;
(b) remove vertices $v$, *if* $v \in C$ and $v$ is connected to less than $p\%(|V(sub(G'))|-|V(C)|)$ vertices;
- repeat Steps 1-4 until no new dense subgraph is discovered in *sub(G')*;

Figure 4 illustrates a clustering example of mining overlapping dense subgraphs. The upper-left graph in the figure is the original graph. Obviously, it has two dense connected subgraphs. The readers can simulate our MODES algorithm to detect these two subgraphs. As demonstrated in this example, MODES is able to discover some dense subgraphs that traditional clustering approaches cannot find.

## 5 COMPARISON WITH OTHER METHODS

Our approach proposed so far simplifies the problem of identifying coherent dense subgraphs across $n$ graphs into a problem of identifying dense subgraphs in two special graphs: the summary graph and the second-order graph. Here, we compare CODENSE with other methods and highlight its major advantages.

*By transforming all necessary information of the n graphs into two graphs, CODENSE achieves significant time and memory efficiency.* Prior frequent graph mining approaches, mostly based on graph pattern expansion technique (Kuramochi & Karypis, 2004; Yan *et al.*, 2005), iteratively extend frequent patterns and check their support. However, these approaches are infeasible in discovering large patterns. The problem of counting the number of distinct maximal frequent subgraphs is #-P-complete, thereby providing a strong implication that the problem of mining maximal frequent subgraphs may be NP-hard (Yang, 2004).

*CODENSE can mine coherent subgraph patterns which likely represent true network modules.* As discussed in the introduction section, frequent patterns may contain sub-

components that differ significantly in their support, while this is not an issue for coherent patterns.

*CODENSE can mine both exact and approximate patterns.* We can control the similarity threshold of the edge support vectors for edge construction in the second-order graph as well as the density threshold for the subgraph discovery in the second-order graph. The lower the two thresholds, the higher the degree of approximation in the output of CODENSE. Conversely, if the support vector similarity threshold is set to be 100% and the 2nd-order dense subgraph is required to form cliques, the identified subgraphs shall occur exactly in the same graphs.

*CODENSE can be extended to pattern mining on weighted graphs* since the core algorithms, normalized-cut and minimum-cut based clustering, can be applied to both weighted and unweighted graphs.

*CODENSE* can be further modified and extended to identify more subtle patterns. For example, instead of clustering edges with overall similar edge support vectors, one may use bi-clustering algorithms (Cheng & Church., 2000) to identify edges showing similar supports in a subset of data sets. Efficient algorithm design for this purpose is currently under development.

# 6 EXPERIMENTAL STUDY

## 6.1 Graph modeling and parameter setting

In this study, we use the co-expression networks derived from microarray data sets as a testing system for CODENSE. We integrated 39 yeast microarray data sets, each comprising the expression profiles of 6,661 genes in at least 8 experiments, from Stanford Microarray Database and the NCBI Gene Expression Omnibus (details of the data set are available at http://zhoulab.usc.edu/CODENSE/). From each microarray data set, we construct a relation network where two genes are connected if they show strong similarity in their expression patterns measured by *Pearson's* correlation. We transform the *Pearson's* correlation (denoted as $r$) into another quantity, $\sqrt{(n-2)r^2/1-r^2}$, and model this quantity as a $t$-distribution with $n$-2 degrees of freedom, where $n$ is the number of measurements used in the computation of the Pearson's correlation.

We construct a summary graph $\hat{G}$, collecting edges with support at least 6 over the 39 relation networks. MODES is first applied to $\hat{G}$ to identify subgraphs with density $\geq d_1$. For each identified subgraph, we then construct a 2nd-order graph by transforming edges to vertices. Here, we build the edge support vector (of length 39) by computing the Pearson's correlation between the expression profiles of two genes (two vertices) in each of the 39 data sets (note that this is different from the binary edge support vectors illustrated in Figure 3). If the Pearson's correlation between two edges' support vectors is significant at $\alpha$=0.001 level, we connect their transformed vertices with an edge in the 2nd-order graph. We again apply the MODES algorithm to the 2nd-order graph to identify subgraphs with density $\geq d_2$. Such identified 2nd-order graph is then transformed back to the summary graph (vertices $\rightarrow$ edges), and MODES is employed once more to identify subgraphs with density $\geq d_3$. In this experiment, we set the three density cutoffs $d_1 = d_2 = d_3 = 0.4$. However, they can be adjusted to accommodate users' specific needs. For example, decreasing $d_1$ and $d_3$ will favor sparse coherent patterns; and increasing $d_2$ will strengthen the co-occurrence of edges across all networks in an identified pattern. In this study, we only focus on the coherent subgraphs with at least four vertices.

## 6.2 Functional module discovery

We applied CODENSE to discover coherent clusters across the 39 co-expression networks and compare the result with the dense subgraphs generated by MODES alone.

To quantify the comparison, we assess the clustering quality by determining the percentage of functionally homogenous clusters among all identified clusters. We used the Gene Ontology (GO) biological process annotation, and consider a cluster to be functionally homogenous if (1) the functional homogeneity modeled by the hypergeometric distribution (Wu *et al.*, 2002) shall be significant at $\alpha$=0.01; and (2) at least 40% of its member genes with known annotations belong to a *specific* GO functional category. Using the GeneOntology biological process annotation, we define specific functions to be those associated with GeneOnology nodes that are more than 5 levels below the root.

We found that CODENSE significantly increased the percentage of functionally homogenous clusters from the MODES results by filtering out larger amounts of noisy genes or noisy clusters. MODES identified 366 clusters, among which 151(42%) are functionally homogenous; Starting with the 366 clusters, after the 2nd-order clustering, CODENSE identified 770 clusters that have at least 4 annotated genes. 76% of these clusters are functionally homogenous. This shows a 34% increase of functionally homogenous clusters compared to the MODES results.

The major performance improvement of CODENSE over MODES is attributed to the power of the 2nd-order clustering in eliminating dense summary subgraphs whose edges do not show co-occurrence across all networks. For example, MODES identified a five-gene clique in the summary graph, {MSF1, PHB1, CBP4, NDI1, SCO2}. However, the five genes come from diverse functional categories such as "protein biosynthesis", "replicative cell aging", and "mitochondrial electron transport". In fact, although all edges of this clique occur in at east 6 networks, their co-occurrence is not significant across the 39 networks (see Figure 5a). The 2nd-order clustering can filter out such pseudo-clusters, therefore provides more reliable results.

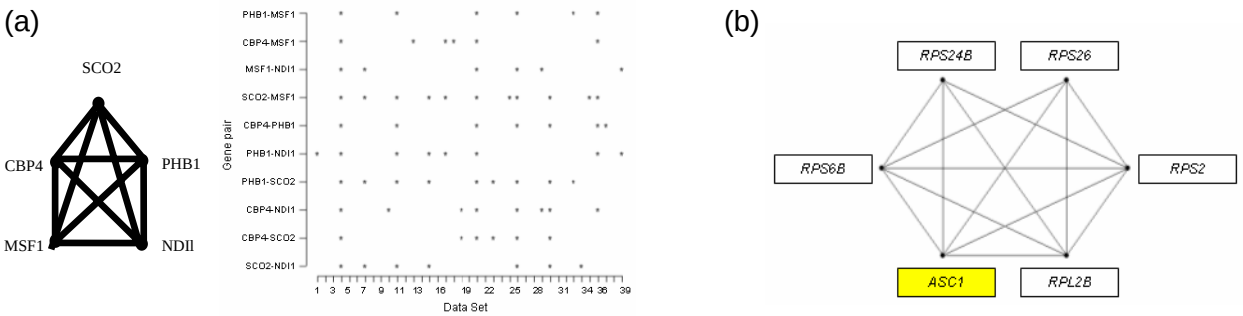


**FIGURE 5.** (a) The edge occurrence profile of the five-gene clique in the summary graph. (b) Shown are the coherent dense subgraph containing 6 genes, all 5 genes except ASC1 are known to be involved in protein biosynthesis. ASC1 is therefore to be predicted to have this function as well.

Moreover, we found instances in which the 2nd-order clustering can remove the noisy genes in a functionally diverse cluster so that its subcluster(s) becomes functionally homogenous. For example, in a cluster containing 54 genes that are associated with diverse functional categories, such as “organic acid metabolism”, “carboxylic acid metabolism", and “amino acid and derivative metabolism”, none of those categories is significantly over-presented in this cluster. After the 2nd-order clustering, a coherent subcluster emerges with 7 annotated genes, which is dominated by the function “organic acid metabolism” (*p*-value=2.18e-05).

### 6.3 Functional annotation

The large number of functionally homogenous clusters identified by CODENSE provides a solid foundation for functional annotation of uncharacterized genes. For those clusters containing unknown genes, if the most dominating GO functional category is significantly over-represented (Bonferroni corrected hypergeometric *P*-value <0.01), we annotated the unknown genes with that function. To assess the prediction accuracy of our method, we employed a “leave-one-out” approach by masking a known gene to be unknown, and assign its function based on the remaining known genes in the cluster. We consider a prediction to be correct, if the lowest common ancestor of the predicted and known functional categories of that gene is 5 levels below the root in the GO hierarchy. That is, the predicted and the known categories will merge into the same category at least at the level 6 of the GO hierarchy. Note that the annotated yeast genes encompass 160 functional categories at the level 6 of the GO hierarchy. We have assigned functions to 448 known genes, and achieved a prediction accuracy of 50%.

By applying this approach to unknown genes, we made functional prediction for 169 genes, covering a wide range of functional categories from cellular protein metabolism, protein biosynthesis, ribosome biogenesis, nucleobase, nucleoside, nucleotide and nucleic acid metabolism, cellular biosynthesis, etc. Figure 5b illustrates an example of our predictions, in which the uncharacterized gene ASC1 is predicted to be involved in “protein biosynthesis”, because all of the remaining 5 genes in the same subgraph participate in that biological process. The comprehensive prediction results are available at http://zhoulab.usc.edu/CODENSE/. Numerous of our predictions are supported by experimental studies in the literature. For example, we predicted RRP15 to participate in “ribosome biogenesis”. According to a personal communication between Fatica and SGD in 2004, this gene is involved in pre-rRNA processing. We assigned the function “protein biosynthesis” to YMR116C; and two studies showed that it is involved in translation regulation (Chantrel *et al.*, 1998) and control (Gerbasi *et al.*, 2004). We predicted QRI5 to be involved in “protein biosynthesis”; QRI5 has been shown to participate in a common regulatory process together with MSS51 (Simon *et al.*, 1992) and the GO annotation of MSS51 is “positive regulation of translation and protein biosynthesis”.

## 7. CONCLUSIONS

We developed a novel algorithm, CODENSE, to efficiently mine coherent dense subgraphs across massive biological networks. In comparison with previous approaches, CODENSE is scalable in the number and the size of the networks to mine, adjustable in terms of exact or approximate coherent pattern mining, and extendable to weighted and directed networks. It provides an efficient tool for the identification of network modules and for the functional discovery in the ever increasing biological networks. The method can integrate heterogeneous network data (e.g., protein interaction network, genetic interaction network, and co-expression networks) to reveal consistent biological signals. The discovered network modules can be used in a variety of biological applications, e.g., predict the func-

tions of unknown genes, construct the transcription modules, and infer the potential protein assembly mechanisms.

## ACKNOWLEDGEMENTS

We thank Min Xu for helpful discussions and Ming-Chih J. Kao for proof-reading of the manuscript. The work of Haiyan Hu is partially supported by the NIH grant 1P50CA112952. The work of Xianghong J. Zhou is supported by the USC faculty setup grant and the NIH grant 5R01GM067243. The work of Xifeng Yan and Jiawei Han is supported by NSF IIS-02-09199.

## REFERENCES

Allocco D.J., Kohane I.S. & Butte A.J. (2004): Quantifying the relationship between co-expression, co-regulation and gene function. *BMC Bioinformatics* **5**, 18.

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

Barabasi A.L. & Oltvai Z.N. (2004): Network biology: understanding the cell's functional organization. *Nat Rev Genet* **5**, 101-13.

Chantrel Y., Gaisne M., Lions C. & Verdiere J. (1998): The transcriptional regulator Hap1p (Cyp1p) is essential for anaerobic or heme-deficient growth of Saccharomyces cerevisiae: Genetic and molecular characterization of an extragenic suppressor that encodes a WD repeat protein. *Genetics* **148**, 559-69.

Cheng Y. & Church. G.M. (2000): Biclustering of Expression Data. *ISMB*, 93-103.

Clare A. & King R.D. (2002): How well do we understand the clusters found in microarray data? *Silico Biology* **2**, 511-22.

Gasch A.P. & Eisen M.B. (2002): Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol* **3**, RESEARCH0059.

Gerbasi V.R., Weaver C.M., Hill S., Friedman D.B. & Link A.J. (2004): Yeast Asc1p and mammalian RACK1 are functionally orthologous core 40S ribosomal proteins that repress gene expression. *Mol Cell Biol* **24**, 8276-87.

Hartuv E. & Shamir R. (2000): A clustering algorithm based on graph connectivity. *Information Processing Letters archive* **76(4-6)**, 175 - 181.

Koyuturk M., Grama A. & Szpankowski W. (2004): An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics* **20 Suppl 1**, I200-I207.

Kuramochi M. & Karypis G. (2004): Finding Frequent Patterns in a Large Sparse Graph. *2004 SIAM Data Mining Conference*.

Lee H.K., Hsu A.K., Sajdak J., Qin J. & Pavlidis P. (2004): Coexpression analysis of human genes across many microarray data sets. *Genome Res* **14**, 1085-94.

Sharan R. & Shamir R. (2000): CLICK: a clustering algorithm with applications to gene expression analysis. *Proc Int Conf Intell Syst Mol Biol* **8**, 307-16.

Shi J. & Malik J. (2000): Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 888--905.

Simon M., Della Seta F., Sor F. & Faye G. (1992): Analysis of the MSS51 region on chromosome XII of Saccharomyces cerevisiae. *Yeast* **8**, 559-67.

Spirin V. & Mirny L.A. (2003): Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A* **100**, 12123-8.

Stoer M. & Wagner F. (1997): A simple min-cut algorithm. *Journal of the ACM* **4**, 585-591.

Van Dongen S. (2000): Graph Clustering by Flow Simulation. *PhD thesis,University of Utrecht.*

Wu L.F., Hughes T.R., Davierwala A.P., Robinson M.D., Stoughton R. & Altschuler S.J. (2002): Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters. *Nat Genet* **31**, 255-65.

Yan X., Zhou X. & Han J. (2005): Mining Closed Relational Graphs with Connectivity Constraints. *Proceedings of the International Conference on Data Engineering*.

Yang G. (2004): The complexity of mining maximal frequent itemsets and maximal frequent patterns. *Proceddings of Int. Conf. on Knowledge Discovery and Data Mining*, 344-353.

Zhou X.J., Kao M.C., Huang H., Wong A., Nunez-Iglesias J., Primig M., Aparicio O.M., Finch C.E., Morgan T.E. & Wong W.H. (2005): Functional annotation and network reconstruction through cross-platform integration of microarray data. *Nat Biotechnol* **23**, 238-43.