Chapter 6
Pattern Mining Across Many Massive Biological Networks

Wenyuan Li, Haiyan Hu, Yu Huang, Haifeng Li, Michael R. Mehan, Juan Nunez-Iglesias, Min Xu, Xifeng Yan, and Xianghong Jasmine Zhou

Abstract The rapid accumulation of biological network data is creating an urgent need for computational methods on integrative network analysis. Thus far, most such methods focused on the analysis of single biological networks. This chapter discusses a suite of methods we developed to mine patterns across many biological networks. Such patterns include frequent dense subgraphs, frequent dense vertex sets, generic frequent patterns, and differential subgraph patterns. Using the identified network patterns, we systematically perform gene functional annotation, regulatory network reconstruction, and genome to phenome mapping. Finally, tensor computation of multiple weighted biological networks, which filled a gap of integrative network biology, is discussed.

1 Introduction

The advancement of high-throughput technology has resulted in the rapid accumulation of data on biological networks. Coexpression networks, protein interaction networks, metabolic networks, genetic interaction networks, and transcription regulatory networks are continuously being generated for a wide-range of organisms under various conditions. This wealth of data represents a great opportunity, to the extent that network biology is rapidly emerging as a discipline in its own right [7, 40]. Thus far, most of the computational methods developed in this field have focused on the analysis of individual biological networks. In many cases, however, a single network is insufficient to discover patterns with multiple facets and subtle signals. There is an urgent need for methods supporting the integrative analysis of *multiple* biological networks.

X.J. Zhou (✉)
Program in Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089, USA
e-mail: xjzhou@usc.edu

M. Koyutürk et al. (eds.), *Functional Coherence of Molecular Networks in Bioinformatics*, DOI 10.1007/978-1-4614-0320-3_6,

Biological networks can be classified into two categories: (1) physical networks, which represent physical interactions among molecules, for example, protein–protein interaction, protein–DNA interaction and metabolic reactions; and (2) conceptual networks, which represent functional associations of molecules derived from genomic data, for example, coexpression relationships extracted from microarray data and genetic interactions obtained from synthetic lethality experiments. While physical networks are still 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 a great deal of valuable information for studying the functions and dynamics of biological systems. Although the methods and experiments described in this chapter are applicable to any type of genome-wide network, we use coexpression networks throughout the chapter due to their abundant availability. We transform each microarray dataset into a coexpression network, where nodes represent genes and the edges can be either weighted or unweighted. In a weighted coexpression network, the edge weights can be coexpression correlations; in an unweighted network, two genes are connected with an edge only if their coexpression correlation is higher than a given threshold. Given k microarray datasets, we can construct k networks with the same node set but different edge sets. We refer to this arrangement as a *relation graph set*, since each network provides information on different relationships among the same set of vertices. Note that in a coexpression network, each gene occurs once and only once. The coexpression networks, therefore, have distinct node labels, and we avoid the NP-hard "subgraph isomorphism problem." We also note that our study is distinct from the body of work on comparing biological networks across species [25, 28–30, 42], where the nodes in different networks can have a many-to-many mapping relationship. The methods described here focus on comparing networks from the same species but generated under different conditions.

This chapter describes several types of patterns that can only be discovered by analyzing multiple graphs, and a set of computational methods designed for mining these patterns. First, we discuss algorithms to identify recurrent patterns in multiple unweighted networks. Next, we define and mine differential patterns in multiple unweighted networks. Finally, we introduce an advanced mathematical model suitable for analyzing multiple weighted networks. We will also show how to use the identified patterns to perform gene function prediction, transcription module reconstruction, and transcriptome to phenome mapping.

2 Mining Recurrent Patterns in Multiple Networks

On account of the noisy nature of high-throughput data, biological networks contain many spurious edges which may lead to the discovery of false patterns. However, since biological modules are active across multiple conditions, we can easily filter out spurious edges by looking for patterns that occur in multiple biological networks. For example, we have demonstrated experimentally that recurrent dense subgraphs in multiple coexpression networks often represent transcriptional and

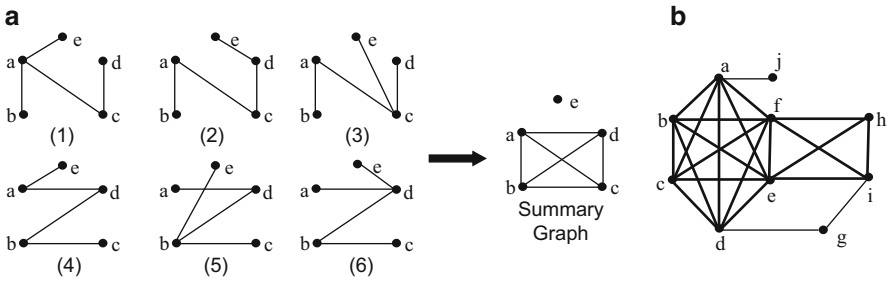


Fig. 6.1 (**a**) Given six graphs with the same vertex set but different edge sets, we construct a summary graph by adding the graphs together and deleting edges that occur fewer than three times. The dense subgraph $\{a,b,c,d\}$ appearing in the summary graph does not occur in any of the original graphs. (**b**) The vertices e and f are shared by cliques $\{a,b,c,d,e,f\}$ and $\{e,f,h,i\}$. The shared vertices can be assigned to both cliques only by approaches that are able to detect overlapping dense subgraphs (cliques are the densest subgraphs of a network)

functional modules [23, 51]. In fact, even recurrent paths are likely to correspond to functional modules [24]. In this section, we define and illustrate three types of recurrent patterns in unweighted graphs, our data mining algorithms to discover them, and their biological applications.

2.1 Coherent Dense Subgraphs

A straightforward approach to analyzing multiple networks is to aggregate these networks together and identify dense subgraphs in the aggregated graph. However, the aggregated graph can contain dense subgraphs that do not occur frequently, or even exist at all, in the original networks. Figure 6.1a illustrates such a case with a cartoon of six graphs. If we add these graphs together to construct a summary graph, we may find a dense subgraph containing vertices a, b, c, and d. Unfortunately, this subgraph is neither dense nor frequent in the original graphs. To overcome this problem, we propose looking for *Coherent Dense Subgraphs* that satisfy two criteria: (1) the nodes are densely interconnected, and (2) all of the edges should exhibit correlated occurrences in the whole graph set. In the following, we provide a formal definition of coherent dense subgraph and an algorithm to identify these patterns in multiple networks.

2.1.1 Problem Formulation

Consider a relation graph set D consisting of n undirected simple graphs: $D = \{G_i = (V,E_i)\}, i = 1,\ldots n, E_i \subseteq V \times V$. All graphs in the set share a common vertex set V. 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, $w_i(u,v) = 1$ if there is an edge between u and v, otherwise $w_i(u,v) = 0$.

Definition 6.1 (Support). Given a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, the support of a graph g is the number of graphs (in D) containing g as a subgraph. This measure is written $support(g)$. A graph is called *frequent* if its support is greater than a specified threshold.

Definition 6.2 (Summary Graph). Given a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, the summary graph of D is an unweighted graph $\hat{G} = (V, \hat{E})$ containing only those edges present in at least k graphs of D. The parameter k is a user-defined support threshold (see an example in Fig. 6.1a).

Definition 6.3 (Edge Support Vector). Given a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, the support vector $\mathbf{w}(e)$ of an edge e is of length n. The ith element of $\mathbf{w}(e)$ is the weight of edge e in the ith graph.

The support vector of edge (a,b) for the six graphs shown in Fig. 6.1a is $[1,1,1,0,0,0]$, while the support vector of edge (b,c) is $[0,0,0,1,1,1]$. Their support vectors clearly show that edges (a,b) and (b,c) are not correlated in this dataset, although both of them are frequent.

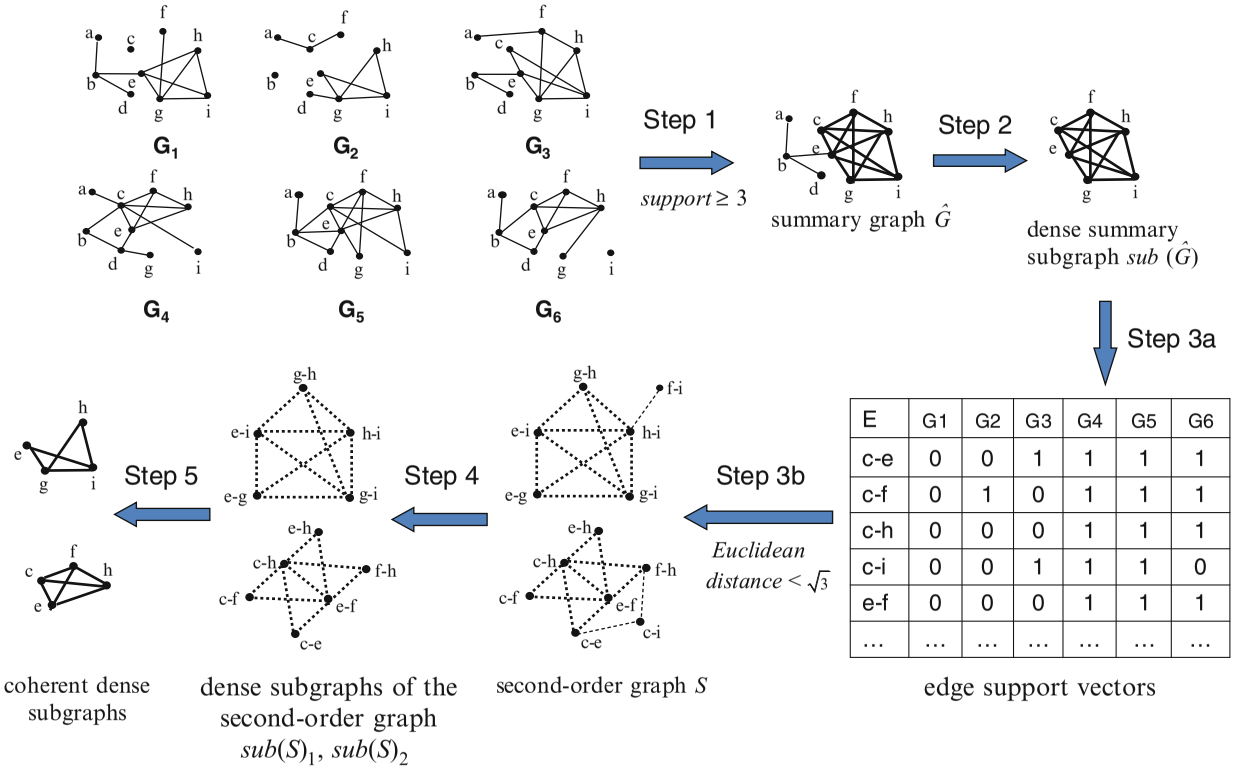
We use a special graph, the *second-order graph* S, to illustrate the co-occurrence of edges in a relation graph set D. Each edge in D is represented as a vertex in S. Two vertices u and v in S are connected if the edge support vectors $w(u)$ and $w(v)$ in D are sufficiently similar. Depending on whether or not the edges in D are weighted, the similarity measure could be the Euclidean distance or Pearson's correlation. Figure 6.2 (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 labeled (c,e) and (c,i) in the second-order graph S. This process is shown in Fig. 6.2. To contrast with the second-order graph, we term the original graphs G_i first-order graphs. This use of the second-order graph is just one type of second-order analysis, a concept proposed in one our previous publications [55].

Definition 6.4 (Second-Order Graph). Given a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, the second-order graph is an unweighted graph $S = (V \times V, E_s)$ whose vertex set is equivalent to the edge set of G. In S, an edge is drawn between vertices u and v if the similarity between the corresponding edge support vectors $\mathbf{w}(u)$ and $\mathbf{w}(v)$ exceeds a specified threshold.

If the first-order graphs G_i are large and dense, S will be impractically large. To more efficiently analyze D, we construct second-order graphs S only for subgraphs of the summary graph $\hat{G}$.

Definition 6.5 (Coherent Graph). Given a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, a subgraph $\text{sub}(\hat{G})$ is coherent if all its edges have support greater than k and if the second-order graph of $\text{sub}(\hat{G})$ is dense.

Definition 6.6 (Graph Density). The density of a graph g, written $density(g)$, is $\frac{2m}{n(n-1)}$, where m is the number of edges and n is the number of vertices in g.



E	G1	G2	G3	G4	G5	G6
c-e	0	0	1	1	1	1
c-f	0	1	0	1	1	1
c-h	0	0	0	1	1	1
c-i	0	0	1	1	1	0
e-f	0	0	0	1	1	1
…	…	…	…	…	…	…

Fig. 6.2 CODENSE: an algorithm to discover coherent dense subgraphs across multiple graphs (the dense subgraphs are marked with *bold* edges)

The problem of mining ***coherent dense subgraphs*** *can now be formulated as follows:* given a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, discover subgraphs g that satisfy the following two criteria: (1) g is a dense subgraph of the summary graph, and (2) g is coherent.

2.1.2 Algorithm

We have developed a scalable algorithm to mine coherent dense subgraphs [23]. It is based on the following two observations concerning the relationship between a coherent dense subgraph, the summary graph, and the second-order graph.

1. If a frequent subgraph of D is dense, then it must also exist as a dense subgraph in the summary graph. However, the converse is not true. A dense subgraph of the summary graph may be neither frequent nor dense in the original dataset (e.g., Fig. 6.1a).
2. If a subgraph is coherent (i.e., if its edges are strongly correlated in their occurrences across a graph set), then its second-order graph must be dense.

These two facts permit the mining of coherent dense subgraphs with reasonable computational cost. According to Observation 1, we can begin our search by finding all dense subgraphs of the summary graph. We can then single out coherent subgraphs by examining their corresponding second-order graphs. Our CODENSE algorithm consists of five steps, as outlined in Algorithm 1 and illustrated in Fig. 6.2. In Steps 2, 4 and 5, we employ a mining algorithm that allows for overlapping dense subgraphs (see Fig. 6.1b).

Step 1. CODENSE builds a summary graph by eliminating infrequent edges.
Step 2. CODENSE identifies dense subgraphs (which may overlap) in the summary graph. Although these dense subgraphs may not be frequently occurring in the original graph set, they are a superset of the true frequent dense subgraphs.
Step 3. CODENSE builds a second-order graph for each dense summary subgraph.
Step 4. CODENSE identifies dense subgraphs in each second-order graph. A high connectivity among vertices in a second-order graph indicates that the corresponding edges have high similarity in their occurrences across the original graphs.
Step 5. CODENSE discovers the real coherent dense subgraphs. Although a dense subgraph $\text{sub}(S)$ found in Step 4 is guaranteed to have the co-occurrent edges in the relation graph set, those edges may not form a dense subgraph in the original summary graph. To eliminate such cases, we convert the vertices in $\text{sub}(S)$ back to edges and apply the overlapping dense subgraph mining algorithm once more. The resulting subgraphs will satisfy both criteria for coherent dense subgraphs: (1) they are dense subgraphs in many of the original graphs, so all of their edges occur frequently; and (2) the support vectors of the edges are highly correlated across the relation graphs.

Algorithm 1: CODENSE

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

2.1.3 Experimental Study

We use coexpression networks derived from 39 yeast microarray datasets as a testing system for CODENSE. Each dataset comprises the expression profiles of 6,661 genes in at least eight experiments. These data were obtained from the Stanford Microarray Database [19] and the NCBI Gene Expression Omnibus [16]. The similarity between two gene expression profiles in a microarray data set is measured by Pearson's correlation. We transform Pearson's correlation (denoted r) into $\sqrt{\frac{(n-1)r^2}{1-r^2}}$, and model the latter quantity as a t-distribution with $n-2$ degrees of freedom (Here, n is the number of measurements used to compute Pearson's correlation). We then construct a relation network for each microarray dataset, connecting two genes if their Pearson's correlation is significant at the $\alpha = 0.01$ level. The summary graph $\hat{G}$ is then constructed by collecting edges with a support of at least six graphs. At all steps where dense subgraph mining is performed (see Algorithm 1), the density threshold is set to 0.4.

To assess the clustering quality, we calculated the percentage of functionally homogeneous clusters among all identified clusters. Based on the Gene Ontology (GO) biological process annotations, we consider a cluster to be functionally homogeneous if (1) the functional homogeneity modeled by the hypergeometric distribution [50] is significant at $\alpha = 0.01$; and (2) at least 40% of its member genes with known annotations belong to a specific GO functional category.

Within the hierarchical organization of GO biological process annotations, we define *specific functions* to be those associated with GO nodes that are more than five levels below the root. CODENSE identified 770 clusters with at least four

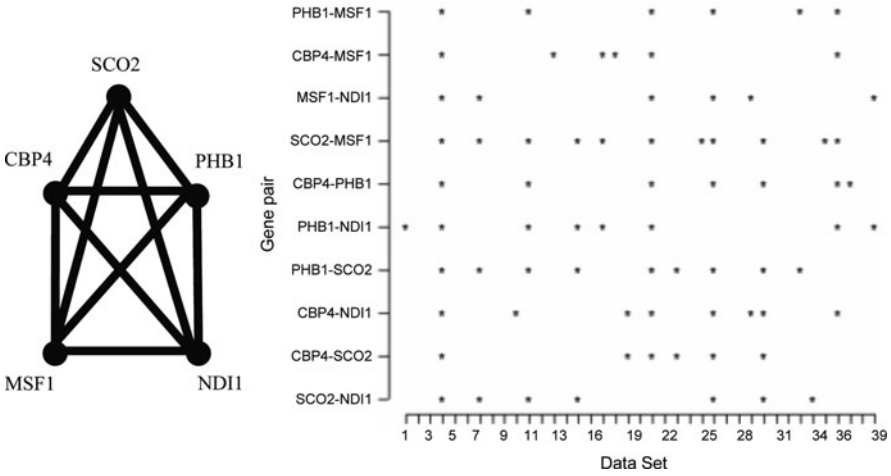


Fig. 6.3 The edge occurrence profiles of a five-gene clique in the summary graph

annotated genes. Of these clusters, 76% are functionally homogeneous. If we stop at Step 2 of the algorithm, obtaining dense subgraphs of the summary graph, only 42% are functionally homogenous. This major improvement in performance can be attributed to the power of second-order clustering as a tool for eliminating dense summary subgraphs whose edges do not show co-occurrence across the networks. As an example, consider the five-gene clique in the summary graph, {MSF1, PHB1, CBP4, NDI1, SCO2}, depicted in Fig. 6.3. The five genes are annotated with a variety of functional categories such as "protein biosynthesis," "replicative cell aging" and "mitochondrial electron transport," so the subgraph is not functionally homogenous. As it turns out, although all edges of this clique occur in at least six networks, their co-occurrence is not significant across the 39 networks (Fig. 6.3). Analyzing the second-order clusters can reveal such pseudoclusters, providing more reliable results.

The large set of functionally homogeneous clusters identified by CODENSE provides a solid foundation for the functional annotation of uncharacterized genes. Some of the clusters contain unknown genes, and if the dominating GO functional category is significantly overrepresented (Bonferroni-corrected hypergeometric p-value < 0.01), we can confidently annotate the unknown genes with that function. To assess the prediction accuracy of our method, we employed a "leave-one-out" approach: a known gene is treated as unknown before analyzing the coherent dense subgraphs, then annotated based on the remaining known genes in the cluster. We consider a prediction correct if the lowest common ancestor of the predicted and known functional categories is five levels below the root in the GO hierarchy. Note that the annotated yeast genes encompass 160 functional categories at level 6 of the GO hierarchy. We predicted the functions of 448 known genes by this method, and achieved an accuracy of 50%. With respect to truly unknown genes, we produced functional predictions for 169 genes, covering a wide-range of functional categories.

2.2 Frequent Dense Vertexset

Although CODENSE has been successfully applied to identify recurrent dense subgraphs across multiple coexpression networks, its criteria are too stringent to identify many potential recurrent coexpression clusters. CODENSE requires coherency of edge recurrence; that is, the entire edge set of a pattern has to show highly correlated recurrence across the graph set. However, edge occurrences in a coexpression network can be distorted by measurement noise or by the correlation threshold used to dichotomize the edges. In fact, any set of genes that is densely connected in a significant number of networks is likely to form a functional and transcriptional module, even if the edges differ from network to network. That is, as long as a consistently large percentage (e.g., $\geqslant 60\%$) of gene pairs in a gene set are connected in multiple networks, that gene set is considered as a recurrently dense pattern and is worthy of attention. We denote such patterns "frequent dense vertexsets" (FDVSs). In this section, we develop a method to efficiently and systematically identify FVDSs.

2.2.1 Problem Formulation

Given a graph $G = (V,E)$ and the subgraph induced by vertex set $V' \subseteq V$, written $G(V')$, we define the FDVS as follows,

Definition 6.7 (Frequent Dense Vertexset). Consider a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$ and each graph shares the vertex set V. Given a density threshold δ and a frequency threshold θ, $V' \subseteq V$ is a frequent dense vertexset if, among all induced graphs $\{G_i(V')\}$, at least $\theta|D|$ graphs have density $\geqslant \delta$.

According to the above definition, a FDVS is a set of vertices, rather than a classical graph with vertices and edges. This definition supports the concept of approximate graph patterns, which need not have exactly the same edge set in the supporting dataset. From a computational point of view, it could be hard to enumerate all of the frequent graphs that satisfy the density constraint. Therefore, we resort to an approximate solution that begins by aggregating the graphs into a summary graph and identifying its dense subgraphs in a top-down manner. The summary graph approach is straightforward, but suffers from two problems: (1) the edges in a dense summary subgraph may never occur together in the original graphs; and (2) noise in the graphs will also accumulate and may become indistinguishable from signals that occur only in a small subset of the graphs. We devised two techniques to overcome these problems. (1) Since similar biological conditions are likely to activate similar sets of transcription/functional modules, we enhance the signal of real patterns by partitioning the input graphs into groups of graphs sharing certain topological properties. Such groups are more likely to contain frequent dense vertexsets. Furthermore, by aggregating similar graphs the signal will be enhanced more than the noise. (2) For each group of graphs, we construct a *neighbor*

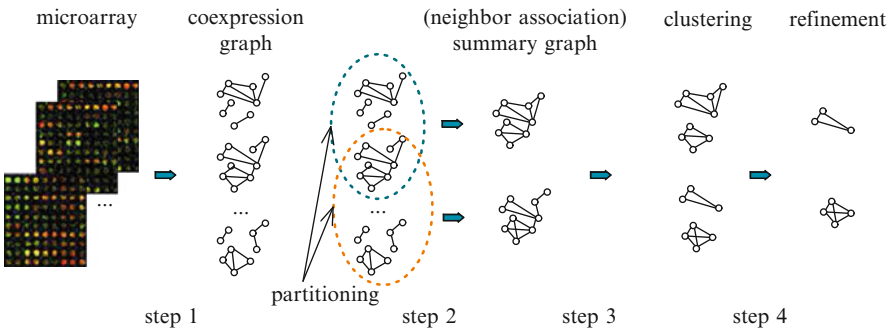


Fig. 6.4 The pipeline of our frequent dense vertexset mining algorithm (called NeMo). Step 1: extract coexpression graphs from multiple microarray datasets by removing insignificant edges. Step 2: partition the coexpression graphs into groups and construct a weighted summary graph for each group. Step 3: cluster each summary graph to identify dense subgraphs. Step 4: refine/extract frequent dense vertexsets from the dense subgraphs discovered in Step 3

association summary graph. This is a weighted graph, unlike the summary graph used by the CODENSE method. The edges of this graph measure the association between two vertices based on their connection strength with their neighbors across multiple graphs. For example, given two vertices u and v, if many small FDVSs include them, these two vertices are likely to belong to the same large FDVS. Figure 6.4 depicts the pipeline of this graph mining methodology. In the next subsection, we will examine this solution in detail.

2.2.2 Algorithm

Given n graphs, a frequent dense vertexset with density δ and frequency θ must form a subgraph with density $\geqslant \delta\theta n$ in the summary graph. According to this observation, we can begin by mining the dense subgraphs of the summary graph. The dense subgraphs are then processed to extract frequent dense vertexsets. This method is outlined as follows:

1. *Construct a summary graph:* Given n graphs, remove infrequent edges and then aggregate all graphs to form a summary graph S.
2. *Mine dense subgraphs from the summary graph:* Apply the overlapping dense subgraph mining algorithm to S. This step yields a set of dense subgraphs $\hat{M}$ satisfying some density constraint, for example, $\geqslant \delta\theta n$.
3. *Refine:* Extract true frequent dense vertexsets from each dense subgraph $\hat{M}$.

For the refinement step, we adopt a heuristic process. Given a dense summary subgraph $\hat{M}$ with n' vertices, we first calculate the weighted sum of the edges incident to each vertex. Next, we sort these n' vertices in ascending order of the

weighted sum. Then, the vertices are removed from the list one by one until the remaining vertices in $\hat{M}$ form a frequent dense vertexset. This approach is referred to as a "greedy" refinement process. More advanced search methods such as simulated annealing can be applied as well. Starting from the dense subgraphs of a summary graph significantly reduces the search space, and provides a good starting point for the refinement process. On the other hand, it could generate false patterns. We offer two improvements to remedy this problem: (1) divide the original graphs into groups and formulate a series of summary graphs based on these groups of graphs. (2) Alter the weights in summary graph to reduce the impact of noisy edges.

To implement the first solution (partitioning the original graph set), we actually begin by mining each individual graph separately for dense subgraphs. The frequent subgraphs are then taken as seed vertexsets to bootstrap the mining process. This bootstrap process is as follows (see Fig. 6.4). (1) Extract dense subgraphs $\hat{M}$ from each individual graph. Then, refine these subgraphs for true frequent dense vertexsets M, using the greedy refinement process introduced above. (2) For each frequent dense vertexset M, calculate its supporting graph set $D_\delta(M) \subseteq D$. Take $D_\delta(M)$ as one subset. (3) Remove duplicate subsets. (4) For each unique subset $D_\delta(M)$, call the summary-graph-based approach introduced above to find frequent dense vertexsets in $D_\delta(M)$.

To implement the second approach (reweighting the summary graph), we introduce the concept of a *neighbor association summary graph*. Its intuition is as follows: given two vertices u and v in a graph, if many small frequent dense subgraphs contain both u and v, it is likely that u and v belong to the same cluster. In other words, if a graph/cluster is dense, then its vertices will share many dense subgraphs. Referring to the definition of a k-vertexlet provided below, let π_u be the set of frequent dense $(k-1)$-vertexlets that contain vertex u, and let $\pi_{u,v}$ be the set of frequent dense k-vertexlets that contain vertices u and v. We also define a scoring function $\text{score}(u,v)$ as follows,

$$\text{score}(u,v) = \frac{|\pi_{u,v}|}{\pi_u} \tag{6.1}$$

Definition 6.8 (Vertexlet). Given a vertex set V, a k-vertexlet is a subset of V with k vertices.

This scoring function is not symmetric: $\text{score}(v,u) \neq \text{score}(u,v)$. We take the average of the two scores, which is symmetric, as the weight of the edge between u and v. This new summary graph is called as the neighbor association graph because it relies on more than one neighbor to determine the weight between two vertices. This weighting method could increase the signal-to-noise ratio for identifying subtle dense subgraphs. The workflow for computing the neighbor association summary graph is outlined in Algorithm 1 of [51]. Once the neighbor association summary graph has been built, we apply the mining routine described above. The entire mining algorithm is named NeMo, for Network Module Mining.

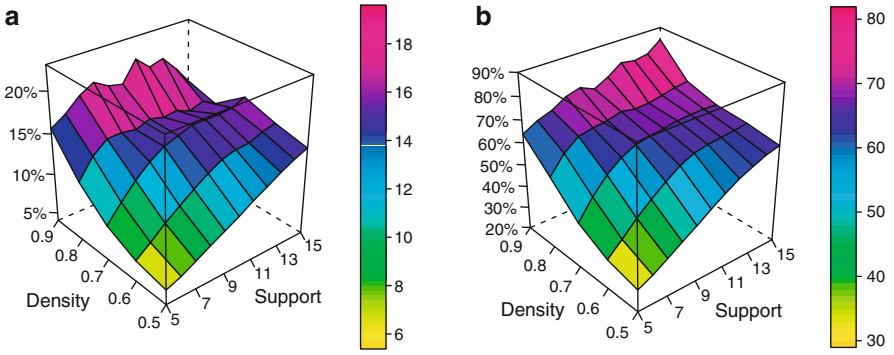


Fig. 6.5 Validation by ChIP-chip and GO data demonstrated that the likelihood of a coexpression cluster being a transcription module and functional homogeneous module increases significantly with its recurrence

2.2.3 Experimental Study

We selected 105 human microarray datasets, generated by the Affymetrix U133 and U95Av2 platforms. Each microarray dataset is modeled as a coexpression graph following the method introduced in Sect. 2.1.3. In this study, the most significant correlations with p-values less than 0.01 (the top 2%) are included in each graph. We applied NeMo to discover frequent dense vertexsets in these networks, and identified 4,727 recurrent coexpression clusters. Each cluster's density is greater than 0.7 in at least ten supporting datasets.

To assess the quality of the clusters identified by NeMo, we tested their member genes for enrichment of the same bound transcription factor. The transcription factors to target gene relationships were ascertained through ChIP-Chip experiments, which contain 9,176 target genes for 20 TFs covering the entire human genome. A recurrent cluster is considered a potential transcriptional module if (1) $>$75% of its genes are bound by the same transcription factor, and (2) the enrichment of the particular TF in the cluster is statistically significant with a hypergeometric p-value $<$0.01 relative to its genome-wide occurrences. Among the identified clusters, 15.4% satisfied both criteria. This is a high hit rate, considering we only tested for 1% of the approximately 2,000 transcription factors estimated to exist in the human genome. On average, the permuted set of clusters was enriched only 0.2% for a common transcription factor. This result demonstrates that our approach can reliably reconstruct regulatory modules. The integrity of the clusters is further validated by varying the threshold for density and recurrence. We find that as these criteria grow stricter, the proportion of identified clusters that share a common bound TF also increases (Fig. 6.5a).

The high quality of the clusters identified by NeMo is also supported by functional homogeneity analysis. We define a cluster to be functionally homogeneous if $>$75% of its member genes belong to the same Gene Ontology biological process with a hypergeometric p-value $<$0.01. As the cluster density and frequency

thresholds increase, the functional homogeneity of the clusters increases as well (Fig. 6.5b). Among all identified clusters, 65.3% are functionally homogeneous compared to 2.2% of the permuted clusters.

2.3 General Recurrent Network Patterns

In the previous two sections, we focused on identifying recurrent dense subgraphs in multiple biological networks. Although such patterns often correspond to functional/transcriptional modules, there also exist many biological modules whose genes are not densely connected. Many types of relationships are possible among functionally-related genes – some lying beyond our current knowledge. These unknowns are exactly the reason why integrative analysis of multiple networks is such a powerful tool. Let us again use coexpression networks as examples. When we combine multiple expression networks, subtle signals may emerge that cannot be identified in any of the individual networks. Such signals include recurrent paths that may extend beyond simple coexpression clusters yet represent functional modules. If we only consider a single coexpression network, it is difficult to stratify functionally important paths from their complex network environment. However, if a path frequently occurs across multiple coexpression networks, it is easily differentiated from the background. In this section, we describe our method to systematically identify recurrent patterns of any kind from multiple relation graphs.

2.3.1 Recurrent Network Pattern Discovery Algorithm

To identify frequently occurring network patterns, we design a data mining procedure based on frequent itemset mining (FIM) and biclustering methods. Given n relation graphs, we wish to identify patterns that comprise at least four interconnected nodes and occur in at least five graphs. This is computationally very difficult due to the large number of potential patterns. Our approach first searches for frequent edge sets that are not necessarily connected, then extracts their connected components. Conceptually, we formulate the n graphs as a matrix where each row represents an edge (i.e., a gene pair), each column represents a graph, and each entry (1 or 0) indicates whether the edge appears in that graph. In this framework, the problem of discovering frequent edge sets can be formulated as a biclustering problem that searches for submatrices with a high density of 1's. This is a well-known NP-hard problem.

We have developed a biclustering algorithm based on simulated annealing to discover frequent edge sets. We employ simulated annealing to maximize the objective function $\frac{c'}{mn+\lambda c}$, where c is the number of 1's in the input matrix, c_0, m and n are the numbers of 1's, rows and columns in the bicluster, respectively, and λ is a regularization factor. Clearly, this objective function favors large biclusters with a high density of 1's. Note that the density is maximized (to unity) when $c' = mn$,

while the size of bicluster is maximized when $c' = c$ (i.e., the pattern is as large as the input matrix). The regularization parameter λ controls the trade-off between density and size. However, there is no theoretical result on suggesting an optimal value for λ. In this study, we tried many heuristic choices of λ. The reported results are based on $\lambda = \frac{0.2}{\max(1,\log_{10}(n_1))}$, where n_1 is the number of edges in the initial configuration (i.e., the seed).

Although this method performs well in our experiments, the enormous search space (the edge/graph matrix has more than 1 million rows and 65 columns) has to be restricted to discover hundreds of thousands of patterns in a reasonable time frame. To address this problem and generate seeds for our biclustering algorithm, we employ the FIM technique [20]. Below, we briefly describe FIM and related concepts.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items and let D be a database of m transactions. Each transaction T is a set of items such that $T \subseteq I$. Supposing X is a set of items $X \subseteq I$, a transaction T is said to contain X if and only if $X \subseteq T$. The itemset X is called frequent if at least s transactions in the database contain X. The output of a standard FIM algorithm is a list of all possible itemsets X which occur in at least s transactions. In our case, we can regard an edge as a transaction and its occurrence in a particular graph as an item. Given our frequency constraint, we need only include edges occurring in at least five graphs in the transaction dataset. Note that the frequent itemsets and their supporting transactions are actually submatrices (biclusters) full of 1's. These clusters with perfect density can serve as seeds for our biclustering algorithm, which searches for larger biclusters that permit holes (i.e., 0's). The FIM algorithm may produce millions of itemsets which contain at least four edges and occur in at least five graphs. These patterns should not be used directly as seeds, however, because they overlap a great deal. This problem is well-known in the data mining community. To improve the seed patterns and reduce unnecessary computation in the biclustering algorithm, we first remove all FIM patterns whose supporting transactions/edges are a subset of some other pattern. Second, we merge two patterns if the resulting submatrix has a density larger than 0.8. This procedure is repeated until no additional merger can happen.

After this postprocessing, we will have about half a million patterns to feed our biclustering algorithm. Given a FIM pattern with v genes, we generate a matrix of all possible edges ($\frac{v(v-1)}{2}$) and all datasets. This matrix serves as a seed for the biclustering algorithm, and is also used as the initial configuration in the algorithm's simulated annealing procedure. Finally, we extract connected components from each output bicluster produced by our algorithm.

2.3.2 Predicting Gene Functions from Recurrent Network Patterns

Given a network pattern, the most popular schemes for predicting gene function employ the hypergeometric distribution to model the probability of genes function based on neighborhood. However, this method ignores the network topology of the

recurrent patterns, which is probably their most important aspect. To avoid this problem, we have developed a new method of estimating gene function based on random walks through the graph that can fully explore the topology of network patterns. Our method is still based on the principle of "guilt by association." In terms of network topology, the degree of association between two genes can be measured by how close they are (i.e., the length of the path between them) and how tightly connected they are (i.e., the number of paths existing between them). Statistically, they translate into the likelihood of reaching one gene from another gene in a random walk. This probability can be approximated by matrix multiplication.

Given a network pattern consisting of v genes, let P be a stochastic matrix of size $v \times v$. The element P_{ij} is $1/n_i$ if genes i and j are connected and 0 otherwise, where n_i is the number of neighbors of gene i. If we regard the genes as states and P_{ij} as the probability of gene/state i transforming into j, then a random walk through the graph can be thought of as a Markov process. From this perspective, it is easy to see that each element of the matrix P^k is the probability that gene i reaches gene j in a k-step random walk. The intuition behind our method is that genes with similar functions are more likely to be well connected (i.e., gene i will reach gene j with high probability in a random walk). Simply put, we expect the probability P^k_{ij} to be large if genes i and j share the same function. Let o be the Gene Ontology binary matrix, where element o_{ij} is 1 if gene i belongs to category j and 0 otherwise. Then, the matrix $M = P^k o$ gives the *network topology scores* of genes relating to functional categories. The higher this score, the more likely a gene has that function. In practice, we choose $k = 3$ to confine our prediction to a local area of network patterns. The function of each gene is estimated as the functional category with the maximum score in the corresponding row of the score matrix M.

In an attempt to improve our method, we tried including attributes other than the network topology scores of a network pattern in the final prediction. These attributes are, recurrence, density, size, average node degree, the percentage of unknown genes, and the functional enrichment of network modules. We use a random forest[1] to determine whether function assignments based on the network topology score are robust. In other words, the purpose of the random forest is to determine whether to accept or reject a functional assignment based on the network topology score. The random forest was trained using the assignments of known genes. The trained model was then applied to classify unknown genes. We only keep the function assignments that the random forest classified as "accept."

2.3.3 Experimental Study

We collected 65 human microarray datasets, including 52 Affymetrix (U133 and U95 platforms) datasets and 13 cDNA datasets from the NCBI Gene Expression Omnibus [16] and SMD [19] databases (December 2005 versions). Each microarray

[1] A random forest is a collection of tree-structured classifiers [8].

dataset is modeled as a coexpression graph following the procedure introduced in Sect. 2.1.3. The FIM and biclustering algorithms described above yield a total of 1,823,518 network patterns (modules) which occur in at least five graphs. After merging patterns with similar network topologies and dataset recurrence, we are left with 143,400 distinctive patterns involving 2,769 known and 1,054 unknown genes. The sizes of the patterns vary from 4 to 180.

We define a module to be functionally homogenous if the hypergeometric p-value after Bonferroni correction is <0.01. Among the identified network patterns, 77.0% are functionally homogenous by this standard. In general, patterns that occur more frequently are more likely to be functionally homogenous. This observation supports our basic motivation for using multiple microarray datasets to enhance functional inferences, namely that by considering pattern recurrence across many networks we can enhance the signal of meaningful structures. We identify network modules with a wide-range of topologies. In fact, 24% of the modules have connectivities <0.5.

To explore the relationships among the network members other than coexpression, we resort to the only available large-scale source: protein interaction data. We retrieved human protein interaction information from the European Bioinformatics Institute (EBI)/IntAct database [21] (version 2006-10-13). For each of the 143,400 detected patterns, we then tested whether protein interactions were overrepresented in member genes compared to all human genes using the hypergeometric test to evaluate significance. A total of 60,556 (22.44%) patterns were enriched in protein interaction at a p-value of 0.001 level. This shows that genes belonging to a module are much more likely to encode interacting proteins. Interestingly, many of the protein-interaction-enriched network modules also fall into functional categories such as protein biosynthesis, DNA metabolism, and so on. There are even many cases where the interacting protein pairs are not coexpressed.

For each of the 143,400 recurrent network patterns, we identified the function of each member gene with the maximum network topology score. We then trained a random forest and made functional predictions for 779 known and 116 unknown genes with 70.5% accuracy. It should be noted that the potential prediction accuracy of this method is probably much higher; the rate of 70% is due to the sparse nature of human GO annotations. Since GO annotations are based only on positive biological evidence, it is likely that many annotated genes have undiscovered functions. Furthermore, the GO directed acyclic graph structure is not perfect.

Since our approach allows a given gene to appear in more than one network module, we are able to perform context-sensitive functional annotation. That is, we can associate each gene multiple functions as well as the network environments in which the gene exerts those functions. These contexts and relationships represent valuable information, even if all of a gene's function are already known. Among our predictions, 20% of genes are assigned multiple functions. This rate is almost certainly an underestimate, since for each network module, we only annotated genes with a single functional category: the one associated with the highest network topology score.

3 Differential Network Patterns

Suppose that a set of biological networks is divided into two classes, for example, those related to a specific disease and those obtained under normal or unrelated conditions. It is then interesting to identify network patterns that differ significantly between these two classes. In fact, it has become clear that many complex conditions such as cancer, autoimmune disease, and heart disease are characterized by specific gene network patterns. Recently, we designed an integrative approach to inferring network modules specific to a phenotype [33]. A series of microarray datasets modeled as coexpression networks is labeled with phenotypic information such as the type of biological sample, a disease state, a drug treatment, etc. For each phenotype, we can partition all microarray datasets into a positive class of datasets appropriately annotated with that phenotype, and a background class containing the rest of the datasets. We have designed a graph-based simulated annealing approach [26] to efficiently identify groups of genes that form dense subnetworks preferentially and repeatedly in a phenotype's positive class. Using 136 microarray datasets, we discovered approximately 120,000 modules specific to 42 phenotypes and developed validation tests combining Gene Ontology, Gene Reference Into Function (GeneRIF) and UMLS data. Our method is applicable to any kind of abundant network data with well-defined phenotype associations, and paves the way for a genome-wide atlas of gene network–phenotype relationships.

3.1 Problem Formulation

Consider a relation graph set $D = \{G_1, G_2, \ldots, G_n\}$, where each graph $G_i = (V, E_i)$ is annotated with a set of phenotypes. For each phenotype, we partition D into a positive class D_{P} consisting of graphs annotated with that phenotype and a background class $D_{\mathrm{P}}^{c} = D \setminus D_{\mathrm{P}}$. Our problem is to identify groups of genes which form dense subgraphs repeatedly in the phenotype positive class but not in the background class. More specifically, we aim to satisfy three criteria: first, a gene set must be densely connected in multiple graphs; second, the annotations of these graphs must be enriched in a specific phenotype; and third, the gene set meeting the first two criteria must be as large as possible. Put simply, this problem is to find modules with three qualities: density, phenotype specificity, and size.

For the first criterion, we consider a gene set to be densely connected if its density is larger than a hard threshold (typically 0.66). However, because we will use simulated annealing as the optimization method (see Sect. 3.2), hard thresholds are too restrictive. Rather, we want the algorithm to accept intermediate states that may be unfavorable. We, therefore, design an objective function f_{dens} with a soft threshold, where unfavorable values of the density increase the cost exponentially. This objective function is defined in (6.3) below. Similarly, the other two criteria also use soft thresholds in their objectives. The second criterion (specificity) states that given a phenotype, we wish to find dense gene sets that occur frequently

in the positive class but infrequently in the background class. The specificity objective function is defined in (6.4). It uses the hypergeometric test to quantify the significance of phenotype enrichment and favors low p-values, again at an exponential rate. For simplicity and computational considerations, we limited the size of the module to 30 genes. We believe this to be an ample margin for phenotypically relevant gene sets. Equation (6.2) shows the size objective function, which contains both a linear component (first term) and an exponential component (second term). The exponential component sets a strong preference for low sizes (four to five vertices), but the linear component continues to reward size increases above this soft threshold.

We supplemented the three main objectives with a fourth: the *density differential* defined in (6.5). This term compliments the density and specificity objective functions by comparing the average density of the cluster in the background datasets to its density in the phenotype datasets. The rationale behind this term is as follows. Since the specificity objective function only takes a state's active datasets as arguments, the transition to a neighboring state may yield a sudden change in the specificity energy because its active datasets are different. However, many neighboring states can have subtle changes in the density distribution among the active and inactive datasets that is not captured by the density and specificity functions alone. The density differential function is, therefore, designed to reward these subtle density changes, helping direct the simulated annealing process toward more phenotype-specific clusters. We found that using the density differential in combination with the specificity and density allowed the algorithm to converge faster and find better clusters than either option alone.

The individual objective functions that we designed take the following forms:

$$f_{\text{size}}(x) = \exp\left\{-\alpha\left(\frac{|x|}{\gamma} - o_s\right)\right\} \tag{6.2}$$

$$f_{\text{dens}}(x) = \exp\left\{-\alpha\left(\min_{i \in D_{\text{A}}}(\delta_i(x)) - o_\delta\right)\right\} \tag{6.3}$$

$$f_{\text{spec}}(x) = \log\left(\mathbb{P}\left(Y \geq |D_{\text{A}} \cap D_{\text{P}}|\right)\right) \tag{6.4}$$

$$f_{\text{diff}}(x) = \left(\frac{1}{|D_{\text{P}}^c|}\sum_{i \in D_{\text{P}}^c}\delta_i(x) - \frac{1}{|D_{\text{P}}|}\sum_{i \in D_{\text{P}}}\delta_i(x)\right) \tag{6.5}$$

where

D_{P} is the set of graphs annotated with the current phenotype,
D_{A} is the set of graphs in which the gene cluster is dense,
and $Y \sim \text{hypergeometric}(|D_{\text{A}}|, |D_{\text{P}}|, |D_{\text{P}}^c|)$.

The exponential components of these functions prevent the simulated annealing algorithm from settling on an extreme case with just one of the desired

qualities (such as a very specific triangle, which is always very dense and small). Improvements to such cases are always rewarded, however, and they are accepted as intermediate steps with good probability. We selected the parameters $\alpha = 20$, $\gamma = 30$, $o_\delta = 0.85$, and $o_s = 0.2$ based on our simulation results comparing biologically validated clusters with clusters arising from random chance.

We combined the four objective functions into a single function using a weighted sum $f(x) = w_1 f_{\text{size}}(x) + w_2 f_{\text{dens}}(x) + w_3 f_{\text{spec}}(x) + w_4 f_{\text{diff}}$. The key difficulty with this approach is determining an appropriate set of weights. In previous studies, this has been accomplished empirically [13]. We do the same, for the following reasons. First, we are interested in finding a single optimal or near-optimal objective function, rather than exploring the extremes of each term. Second, the overall effectiveness of our algorithm turns out to be consistent for a wide-range of weights. Finally, although we chose weights based on the algorithm's performance with simulated data, it also behaved well on real data. The weights for size, density and specificity, and density differential are 0.05, 0.05, 5, and 50, respectively.

3.2 *Differential Network Pattern Discovery Algorithm*

As stated above, we use simulated annealing (SA) to identify differential patterns. This well-established stochastic algorithm has been successfully applied many other NP-complete problems [44]. Our specific design for the SA algorithm follows.

3.2.1 Search Space

A *state* is defined as a set of vertices, and the search space is the set of all possible states. For simplicity and computational considerations, we limit the space to sets with fewer than 30 vertices. We believe this to be an ample margin for phenotypically relevant gene sets. Formally, we define the search space as $\mathscr{S} = \{x : x \subset V, |x| \leq 30, |x| \geq 3\}$.

3.2.2 Differential Coexpression Graphs

To dramatically increase the probability of finding optimal modules across many massive networks, we wish to narrow down the search space. We, therefore, construct a weighted *differential coexpression graph* for each phenotype. This graph summarizes the differences between gene coexpression networks in the phenotype class and those in the background class. The differential coexpression graph is used by the SA algorithm to create neighboring states (see Sect. 3.2.4).

The weighted differential coexpression graph $G_\Delta = (V, E_\Delta)$ contains only edges (coexpression relationships) that are present frequently in D_{P} but infrequently in D_{P}^c. The specificity of a single edge can be measured by the significance p of

a hypergeometric test comparing the abundance of the edge in D_P to its overall abundance in D. The vertex set V of G_Δ is the same as that of D, and the weight of an edge is $-\log(p)$. In this way, heavier edges in this graph represent pairs of genes that exhibit elevated coexpression highly specific to D_P.

3.2.3 Initial States

SA attempts to find a global optimum state. If we were to use random initial states and run the algorithm for a long time, we will always arrive at approximately the same final state: the largest vertex set having the most evidence for coexpression and phenotype specificity. However, we are interested in finding many independent vertex sets. We, therefore, designed a systematic way of generating initial states ("seeds") and restricted the SA search space to vertex sets containing these seeds.

We define a *triangle* as a set of three vertices that is fully connected in at least one dataset. The hypothesis underlying our strategy is that if a set of genes is coexpressed specifically in datasets annotated with the phenotype of interest, then this set will include at least one triangle that appears frequently in the positive class and rarely in the background class.

Therefore, for each phenotype we tested every triangle appearing in the positive class for enrichment (using the hypergeometric test) with respect to the background class. For each triangle with a hypergeometric p-value less than 0.01, we ran the SA algorithm with the constraint that states must be supersets of the initial triangle.

3.2.4 Selection of Neighboring States

We define a *neighbor* of the current state as any state containing either one more or one fewer vertex. We create neighboring states by first determining whether to add or remove a vertex, then choosing the vertex based on an appropriate probability distribution.

If a cluster has size 3, it consists only of the initial seed so a vertex must be added. If a cluster has size 30 (maximum), a vertex must be removed. For intermediate values, we proceed as follows.

Let x be the current state. We narrow the choice of vertices to be added by considering only those with at least one edge to a vertex in x in at least one of the phenotype datasets. This criterion is easily justified, as no other vertices could possibly contribute to x as a dense, phenotype-specific cluster, even as an intermediate step. It can be shown that this set corresponds exactly to $\mathcal{N}_x = \{g : g \notin x, \sum_{h \in x} w_\Delta(g,h) > 0\}$ (See Sect. 3.2.2).

The probability of removing a vertex is given by $p_{\text{rem}} = s_0/|\mathcal{N}_x|$, where s_0 is an estimate of how many vertices will improve the state. This simple function allows the SA process ample time to consider many neighbors before attempting to remove a vertex, since the number of neighboring vertices vastly outnumbers the number

of vertices in a cluster. We heuristically chose $s_0 = 20$ as an appropriate average number. In the future, an iterative estimation of s_0 as the average size of the returned clusters might improve the performance of the algorithm.

In the event that a gene is to be removed, it is chosen uniformly from the cluster. When adding a gene, however, the probability of selecting vertex $g \in \mathcal{N}_x$ is proportional to the summed weights of edges in the differential coexpression graph leading from g to members of x. Formally, we have: $\mathbb{P}(g_a \text{ is added}) = \sum_{a \in x} w_\Delta(g_a, a) / \sum_{b \in \mathcal{N}_x} \sum_{a \in x} w_\Delta(a, b)$.

3.2.5 Annealing Schedule

We used the schedule $T_k = T_{\max}/\log(k+1)$, where k is the iteration number and T_k is the temperature at that iteration [18]. The initial temperature for our study was 4. This schedule form guarantees optimality for long run times. Although it might be argued that long run times are impractical, we found that for an identical number of iterations, this schedule resulted in lower-energy clusters than the oft-used exponential schedule $T_{k+1} = \alpha T_k = \alpha^k T_{\max}$. We ran the algorithm for a maximum of 1,000,000 iterations or until the simulated annealing converged. In cases where the maximum number of iterations was reached, we forced convergence to the best local minimum by a near-greedy exploration of the neighborhood, achieved by decreasing the temperature to near zero.

3.2.6 Postfiltering

Recall that we forced the initial seed triangle to be part of the final result. Clearly, some of these seeds will result from noise alone, in which case the final output will not be biologically significant. To remove these clusters, we discarded any vertex set not meeting the following criteria: size greater than 6, density greater than 0.66, and FDR-corrected phenotype specificity (p-value) less than 0.01. Moreover, the cluster must be dense in at least three datasets related to the target phenotype. After filtering, we merged redundant clusters with intersections/unions greater than 0.8.

3.3 Experimental Study

We selected microarray datasets from NCBI's Gene Expression Omnibus [16] that met the following criteria: all samples were of human origin, the dataset had at least eight samples (a minimum for accurate correlation estimation), and the platform was either GPL91 (Affymetrix HG-U95A) or GPL96 (Affymetrix HG-U133A). Throughout this study, we only considered the 8,635 genes shared by both platforms (and therefore all datasets). All 136 datasets meeting these criteria on 28 Feb 2007 were used for the analysis described herein.

We determined the phenotypic context of a microarray dataset by mapping the Medical Subject Headings (MeSH) of its PubMed record to UMLS concepts. This process is more refined than scanning the abstract or full text of the paper, and in practice results in much cleaner and more reliable annotations [9, 10]. UMLS is the largest available compendium of biomedical vocabulary, with definitions and hierarchical relationships spanning approximately one million interrelated concepts. The UMLS concepts include diseases, treatments, and phenotypes at various levels of resolution (molecules, cells, tissues, and whole organisms). To infer higher-order links between datasets, we annotated each dataset with all matching UMLS concepts and their ancestor concepts. The datasets received a total of 467 annotations, of which 80 mapped to more than five datasets. Some of the latter were mapped to identical sets of datasets; after merging these, we were left with 60.

For each dataset, we used the Jackknife Pearson correlation as a measure of similarity between two genes (the minimum of the leave-one-out Pearson correlations). To create the coexpression network, we selected a cutoff corresponding to the 150,000 strongest correlations (0.4% of the total number of gene pairs: $\binom{8,635}{2} \approx 3.73 \times 10^7$). This choice was motivated by exploring the statistical distribution of pairwise correlations, which we do not detail here.

We applied our simulated annealing approach to all 136 microarray datasets covering 42 phenotype classes. The phenotypes related to a wide-range of diseases (e.g., leukemia, myopathy, and nervous system disorders) and tissues (e.g., brain, lung, and muscle). The procedure described above identified 118,772 clusters that satisfied our criteria for a concept-specific coexpression cluster. The number of clusters found for a given phenotype increased with the number of datasets annotated with that phenotype: most of the phenotypes with only a few associated datasets yielded few clusters. The most strongly represented phenotype was "nervous system disorders," with 15 associated datasets and 22,388 clusters.

We used two different methods to evaluate cluster quality. First, we assessed the functional homogeneity of a cluster by testing for enrichment for specific Gene Ontology [14] biological process terms. If a cluster is enriched in a GO term with a hypergeometric p-value less than 0.01, we consider it functionally homogeneous. Of the 118,772 clusters derived from all phenotypes, 78.98% were functionally homogenous. This validation demonstrates a key advantage of our approach: by focusing on clusters specific to a phenotypically related subset of all datasets, we are less likely to detect constitutively expressed clusters such as those consisting of ribosomal genes or genes involved in protein synthesis.

While the GO database provides information on a gene's functions, it fails to describe its phenotypic implications. To map individual genes to phenotypes, we used GeneRIF [34]. This database contains short statements derived directly from publications describing the functions, processes, and diseases in which a gene is implicated. We mapped the GeneRIF notes to UMLS metathesaurus terms (as with the dataset MeSH headings), then annotated genes with the UMLS concepts. Similar

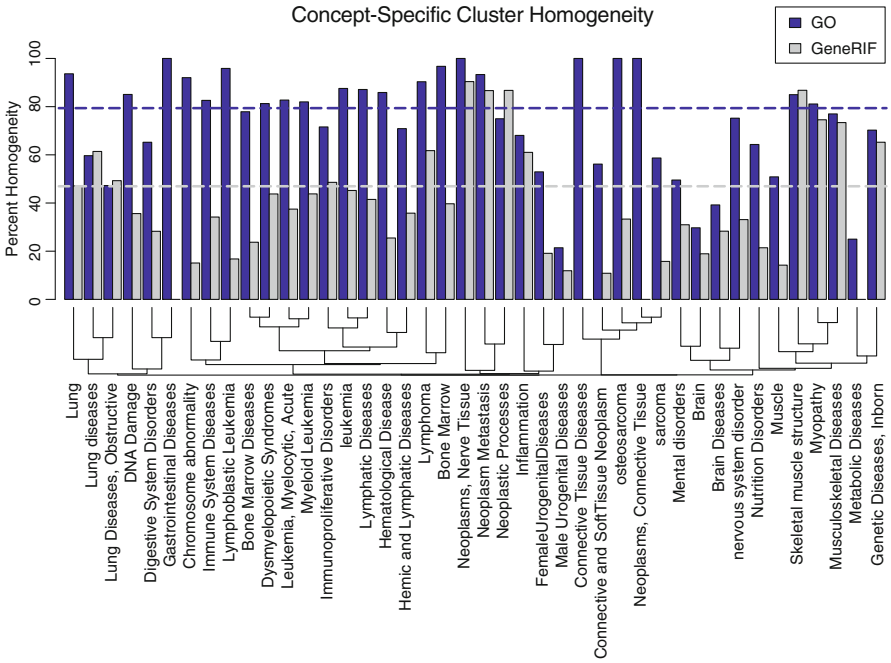


Fig. 6.6 Cluster homogeneity by phenotype. For each phenotype, the proportion of clusters that are significantly enriched (p-value <0.01) for a GO biological process (*blue*) or a GeneRIF UMLS concept (*gray*). The *dotted lines* show the overall homogeneity for all clusters. The dendrogram shows the distance between phenotypes in terms of dataset overlap

to our analysis of the GO annotations, we then assessed the *conceptual homogeneity* of gene clusters in specific UMLS keywords with the hypergeometric test, enforcing a p-value of 0.01 or less. The proportion of conceptually homogeneous modules was 48.3%. Clusters are less likely to have conceptual homogeneity than functional homogeneity, probably due to a dearth of GeneRIF annotations. In some situations, however, GeneRIF performs better. For example, many cancer-related phenotypes such as "Carcinoma," "Neoplasm Metastasis," and "Neoplastic Processes" are more likely to have GeneRIF homogeneity. This effect could be attributed to the abundance of related literature. The functional and conceptual homogeneity of clusters derived from different phenotype classes is summarized in Fig. 6.6.

In addition to testing for functional and conceptual homogeneity, we assessed whether the clusters were involved in the phenotype condition in which they were found. Again, we used both GO and GeneRIF independently for this.

Recall that each functionally homogeneous module is associated with one or more GO biological functions, and also with the phenotype in which it was found. We summarize the GO functions by mapping them to "informative nodes," a concept

we introduced in our earlier work [54]. We then tested them for overrepresentation in that phenotype class. This provided, for each of 33 phenotypes (out of 42 phenotypes having at least one module), a list of gene module functions that are active in that phenotype more often than expected by chance. Many of these GO functions are clearly related to the phenotype in which they were found. For example, the phenotype "Mental disorders" has three GO biological processes related to brain function: "synaptic transmission" (2.3e–62), "neuron differentiation" (5.4e–42), and "central nervous system development" (7.9e–25). Our approach also identifies biological processes related to tissue phenotypes. For example, the "Skeletal muscle structure" phenotype is significantly enriched with modules that are homogeneous in the biological functions "muscle system process" (4.0e–221), "actin filament-based process" (1.23e–150), and "skeletal development (1.53e–03)." The functional association between a module's GO function and the phenotype in which it is active suggests that our clusters are indeed linked to the phenotype conditions in which they were identified. In addition to GO informative nodes, we also tested each phenotype for overrepresentation of UMLS concepts from GeneRIF. This overrepresentation shows which diseases, tissues, and biological concepts are significantly enriched in each phenotype. In Table 6.1, we highlight some of these overrepresented functions and concepts.

The preceding analysis relies on our subjective evaluation of matches between UMLS and GO terms. We can conduct a more objective analysis using the GeneRIF data, which can be mapped directly to the same UMLS terms used to classify phenotypes. We counted the modules that were conceptually homogeneous with respect to the UMLS annotations that defined their respective phenotype classes. Of the 42 phenotypes represented in our study, 26 had one or more matching modules. The proportion of matching modules among total modules in these 26 phenotypes ranged from 0.04 to 33.6%. Although these numbers may not sound impressive, these proportions are significantly larger than expected by chance. We used a permutation test to assess the statistical significance of our analysis. We randomly assigned existing clusters to one of the 47 phenotypes with at least one cluster, while holding the number of clusters assigned to each phenotype constant. One million of these permutations were generated. Thirteen of the phenotypes were found to be significantly enriched with conceptually homogenous modules after FDR correction. They are shown in Table 6.2. The high significance for many of the phenotypes indicates that the low percentages are probably due to a dearth of GeneRIF annotations. As GeneRIF becomes more comprehensive, we expect the performance to improve in both the percentage of matching clusters and the number of phenotypes that are significant. We also found that the UMLS text mining of the GeneRIF database and the MeSH headers is not perfect, so improvements and refinements in those areas should also improve our validation results.

Table 6.1 Selected UMLS Concepts and their principal annotations. We annotated clusters using Gene ontology and GeneRIF, as detailed in the text. We then identified the annotations that were preferentially found in one concept relative to the others, as assessed by the hypergeometric test (Bonferroni-corrected *p*-values shown in parentheses)

Concept	Total	Over-represented GO annotations	Over-represented GeneRIF annotations
Lymphoma	890	Cell cycle phase (9.2e–276)	Lymphoreticular tumor (2.6e–93)
		Cell cycle checkpoint (1.2e–14)	Abnormal Hematopoietic and lymphoid cell (2.6e–22)
		Regulation of cell cycle process (3.2e–08)	Low grade B-cell lymphoma morphology (3.5e–19)
		Antigen processing and presentation (7.7e–03)	
Mental disorders	866	Synaptic transmission (2.3e–62)	Schizophrenia (4.3e–12)
		Neuron differentiation (5.4e–42)	Neurons (1.2e–11)
		Central nervous system development (7.9e–25)	Alzheimer's disease (3.4e–04)
Muscle	584	Muscle system process (7.9e–52)	Heart (1.2e–20)
			Intrathoracic cardiovascular structure (3.1e–19)
			Muscle, striated (8.2e–15)
Myopathy	6,328	Actin filament-based process (7.2e–21)	Coronary heart disease (<1e–324)
		Muscle system process (4.6e–06)	Disorder of skeletal muscle (<1e–324)
Neoplastic processes	1,486	Keratinocyte differentiation (<1e–324)	Lung neoplasms (2.6e–207)
		Cell cycle checkpoint (1.0e–124)	Triploidy and polyploidy (2.8e–179)
		Regulation of mitotic cell cycle (7.4e–122)	Tumor of dermis (8.2e–123)
		Cell division (1.7e–107)	Glioma (6.4e–121)
Skeletal muscle structure	6,719	Muscle system process (4.0e–221)	Musculoskeletal structure of limb (4.3e–46)
		Actin filament-based process (1.2e–150)	Heart (7.6e–46)
		Skeletal development (1.5e–03)	

Table 6.2 Phenotypes for which the annotated clusters are consistent with the phenotype class in which they were derived. The first column indicates a UMLS phenotype. The second column displays the total number of clusters active in that phenotype class. The third and fourth columns show the percentage of clusters annotated with that phenotype in the phenotype class and in the background class, respectively. The fifth column shows the FDR-corrected *p*-value for the difference between the classes. The statistical significance was calculated by permuting the clusters across the dataset phenotypes 1,000,000 times. Concepts with a *p*-value less than 4.7e–6 were never outperformed by the permutations

Phenotype	Total clusters in phenotype class	Matching clusters in phenotype class (%)	Matching clusters in background class (%)	*p*-value
Mental disorders	791	3.12	0.17	<4.7e–06
Lymphoma	409	20.11	0.97	<4.7e–06
Myopathy	645	15.46	3.65	<4.7e–06
Musculoskeletal diseases	1,619	2.26	1.33	<4.7e–06
Genetic diseases, inborn	1,470	7.86	1.82	<4.7e–06
Neoplasms, nerve tissue	765	33.60	2.02	<4.7e–06
Neoplastic processes	794	9.08	4.19	<4.7e–06
Nervous system disorder	2,214	4.44	2.69	<4.7e–06
Skeletal muscle structure	154	0.94	0.18	<4.7e–06
Hemic and lymphatic diseases	1,129	1.17	0.65	1.3e–05
Bone marrow diseases	523	1.31	0.52	5.3e–03
Leukemia	460	0.55	0.36	2.9e–02
Muscle	483	1.03	0.31	3.5e–02

4 A Computational Model for Multiple Weighted Networks: Tensor

In previous sections, we approached the analysis of multiple large networks through a series of heuristic, graph-based, data mining algorithms. While useful, this class of methods faces two major limitations. (1) The general strategy is a stepwise reduction of the large search space, but each step involves one or more arbitrary cutoffs. In addition, there is the initial cutoff that transforms continuous measurements (e.g., expression correlations) into unweighted edges. The ad hoc nature of these cutoffs has been a major criticism directed at this body of work. (2) These algorithms cannot be easily extended to weighted networks. Most graph-based approaches to multiple network analysis are restricted to unweighted networks, partly because weighted networks are often perceived as harder to analyze [36]. However, weighted networks are obviously more informative than their unweighted counterparts. Generating an unweighted network by applying a threshold to weighted edges invariably leads to information loss [41]. Furthermore, if there is no reasonable way to choose the

threshold, this loss cannot be controlled. Both problems justify the development of an efficient computational framework suitable for mining patterns in many large weighted networks.

Generally speaking, a network of n vertices can be represented as $n \times n$ adjacency matrix $A = (a_{ij})_{n\times n}$, where each element a_{ij} is the weight of the edge between vertices i and j. A number of numerical methods for matrix computation have been elegantly applied to network analysis, for example, graph clustering [12, 15, 31, 37] and pathway analysis [5, 6]. In light of these successful applications, we propose a tensor-based computational framework capable of analyzing multiple weighted and unweighted networks in an efficient, effective, and scalable manner.

Simply put, a tensor is a multi-dimensional array and a matrix is a second-order tensor. Given m networks with the same n vertices but different topologies, we can represent the whole system as a third-order tensor $\mathscr{A} = (a_{ijk})_{n\times n\times m}$. Each element a_{ijk} is the weight of the edge between vertices i and j in the kth network. By representing a set of networks in this fashion, we gain access to a wealth of numerical methods – in particular continuous optimization methods. In fact, reformulating discrete problems as continuous optimization problems is a long-standing tradition in graph theory. There have been many successful examples, such as using a Hopfield neural network for the traveling salesman problem [22] and applying the Motzkin–Straus theorem to solve the clique-finding problem [35].

Continuous optimization techniques offer several advantages over discrete pattern mining methods. First, we may discover unexpected theoretical properties that would be invisible in a purely discrete analysis. For example, Motzkin and Straus's continuous formulation of the clique-finding problem revealed some remarkable and intriguing properties of cliques which directly benefit this work. Second, when a graph pattern mining problem is transformed into a continuous optimization problem, it becomes easy to incorporate constraints representing prior knowledge. Finally, advanced continuous optimization techniques require very few ad hoc parameters. Although tensor analysis has been productively applied in the fields of psychometrics [11, 49], image processing and computer vision [3, 48], chemometrics [43], and social network analysis [1, 27], this approach has been explored only recently in large-scale data mining [17, 32, 45–47] and bioinformatics [2, 4, 38].

In this section, we develop a tensor-based computational framework to analyze multiple weighted networks by generalizing the problem of finding heavy subgraphs in a single weighted network. A *heavy subgraph* (HS) is a subset of nodes which are heavily interconnected. We extend this concept to multiple weighted networks. By defining a *recurrent heavy subgraph* (RHS) as a subset of nodes which are heavily interconnected in a subset of weighted networks with identical nodes but different topologies. A RHS can be intuitively understood as HS that appears in multiple networks. The nodes of the RHS are always the same, although the weights of the edges may vary between networks (Fig. 6.7).

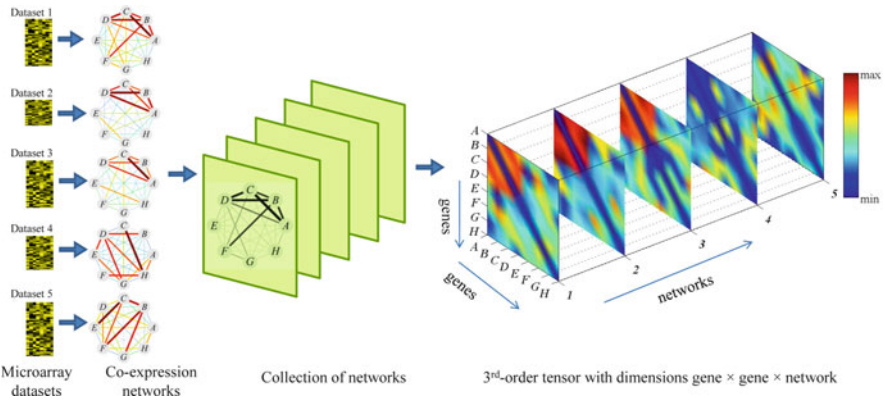


Fig. 6.7 A collection of coexpression networks can be "stacked" together into a third-order tensor such that each slice represents the adjacency matrix of one network. The weights of edges in the coexpression networks and their corresponding tensor elements are indicated by the color scale to the right of the figure. After reordering the tensor using the gene and network membership vectors, it becomes clear that the subtensor in the *top-left* corner of the tensor (formed by genes A, B, C, D in networks $1, 2, 3$) corresponds to a recurring heavy subgraph

4.1 Problem Formulation and Optimization Algorithm

Given m networks with the same n vertices but different topologies, we can represent the whole system as a third-order tensor $\mathscr{A} = (a_{ijk})_{n\times n\times m}$. Each element a_{ijk} is the weight of the edge between vertices i and j in the kth network. The genes and networks forming an RHS are described by two membership vectors: (1) the *gene membership vector* $\mathbf{x} = (x_1, \ldots, x_n)^T$, where $x_i = 1$ if gene i belongs to the RHS and $x_i = 0$ otherwise; and (2) the *network membership vector* $\mathbf{y} = (y_1, \ldots, y_m)^T$, where $y_j = 1$ if the RHS appears in the network j and $y_j = 0$ otherwise. The summed weight of all edges in the RHS is

$$H_{\mathscr{A}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} a_{ijk}x_i x_j y_k \tag{6.6}$$

Note that only the weights of edges a_{ijk} with $x_i = x_j = y_k = 1$ are counted in $H_{\mathscr{A}}$. Thus, $H_{\mathscr{A}}(\mathbf{x}, \mathbf{y})$ measures the "heaviness" of the network defined by $\mathbf{x}$ and $\mathbf{y}$.

To identify a RHS of K_1 genes and K_2 networks intuitively, we should look for the binary membership vectors $\mathbf{x}$ and $\mathbf{y}$ that jointly maximize $H_{\mathscr{A}}$ under the constraints $\sum_{i=1}^{n} x_i = K_1$ and $\sum_{j=1}^{m} y_j = K_2$. This cubic integer programming problem is NP-hard [39]. We instead seek an efficient polynomial solution by reformulating

the task as a continuous optimization problem. That is, we look for real vectors $\mathbf{x}$ and $\mathbf{y}$ that jointly maximize $H_{\mathscr{A}}$. This optimization problem is formally expressed as follows:

$$\begin{array}{l} \max_{\mathbf{x}\in\mathbb{R}_+^n,\mathbf{y}\in\mathbb{R}_+^m} \quad H_{\mathscr{A}}(\mathbf{x},\mathbf{y}) \\ \text{subject to} \begin{cases} f(\mathbf{x}) = 1 \\ g(\mathbf{y}) = 1 \end{cases} \end{array}, \tag{6.7}$$

where $\mathbb{R}_+$ is a nonnegative real space, and $f(\mathbf{x})$ and $g(\mathbf{y})$ are vector norms. This formulation describes a tensor-based computational framework for the RHS identification problem. By solving (6.7), users can easily identify frequent heavy subgraphs consisting of the top-ranking networks (after sorting the tensor by $\mathbf{y}$) and top-ranking genes (after sorting each network by $\mathbf{x}$). After discovering the heaviest RHS in this manner, we can mask it with zeros and optimize (6.7) again to search for the next heaviest RHS.

Two major components of the framework described in (6.7) remain to be designed: (1) the vector norm constraints $(f(\mathbf{x}), g(\mathbf{y}))$, and (2) a protocol for maximizing $H_{\mathscr{A}}(\mathbf{x},\mathbf{y})$. We explain our design choices below.

4.1.1 Vector Norm Constraints

The choice of vector norms will significantly impact the outcome of the optimization. The norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is typically defined in the form $\|\mathbf{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$, where $p \geqslant 0$. The symbol $\|\mathbf{x}\|_p$, called the "L_p-vector norm," refers to this formula for the given value of p. In general, the L_0 norm leads to sparse solutions where only a few components of the membership vectors are significantly different from zero [52]. The L_∞ norm generally gives a "smooth" solution where the elements of the optimized vector are approximately equal.

In our problem, a RHS is a subset of genes that are heavily connected to each other in as many networks as possible. These requirements can be encoded as follows. (1) *A subset of values in each gene membership vector should be significantly nonzero and close to each other, while the rest are close to zero*. To this end, we consider the mixed norm $L_{0,\infty}(\mathbf{x}) = \alpha\|\mathbf{x}\|_0 + (1-\alpha)\|\mathbf{x}\|_\infty$ $(0 < \alpha < 1)$ for $f(\mathbf{x})$. Since L_0 favors sparse vectors and L_∞ favors uniform vectors, a suitable choice of α should yield vectors with a few nonzero significant elements that are similar in magnitude, while all other elements are close to zero. In practice, we approximate $L_{0,\infty}$ with the mixed norm $L_{p,2}(\mathbf{x}) = \alpha\|\mathbf{x}\|_p + (1-\alpha)\|\mathbf{x}\|_2$, where $p < 1$. (2) *As many network membership values as possible are nonzero and close to each other*. As discussed above, this is the typical outcome of optimization using the L_∞ norm. In practice, we approximate L_∞ with $L_q(\mathbf{y})$ where $q > 1$ for $g(\mathbf{y})$. In our experiments, we tested several different settings and finally settled on $p = 0.8$, $\alpha = 0.2$, and $q = 10$ as effective choices for discovering a RHS.

4.1.2 Multi-Stage Convex Relaxation Optimization

Our tensor framework requires an effective optimization method that can deal with nonconvex constraints. It is well-known that the global optimum of a convex problem can be easily computed, while the quality of the optimum for a nonconvex problem depends heavily on the numerical procedure. Standard numerical techniques such as gradient descent lead to a local minimum of the solution space, and different procedures often find different local minima. Considering the fact that most sparse constraints are nonconvex, it is important to find a theoretically justified numerical procedure that leads to a *reproducible solution*.

We use our previously developed framework, known as Multi-Stage Convex Relaxation (MSCR) [52, 53], to design the optimization protocol. MSCR has good statistical properties, and has been proven to generate reproducible solutions even for nonconvex optimization problems [52, 53]. In this context, concave duality will be used to construct a sequence of convex relaxations that give increasingly accurate approximations to the original nonconvex problem. We approximate the sparse constraint function $f(\mathbf{x})$ by the convex function $\widetilde{f}_{\mathbf{v}}(\mathbf{x}) = \mathbf{v}^T h(\mathbf{x}) + f_h^*(\mathbf{v})$, where $h(\mathbf{x})$ is a specific convex function $h(x) = x^h$ ($h \geqslant 1$) and $f_h^*(\mathbf{v})$ is the concave dual of the function $\overline{f}_h(\mathbf{v})$ (defined as $f(\mathbf{v}) = \overline{f}_h(h(\mathbf{v}))$). The vector $\mathbf{v}$ contains coefficients that will be automatically generated during the optimization process. After each optimization, the new coefficient vector $\mathbf{v}$ yields a convex function $\widetilde{f}_{\mathbf{v}}(\mathbf{x})$ that more closely approximates the original nonconvex function $f(\mathbf{x})$.

4.2 Experimental Study

We applied our methods to 129 microarray datasets generated by different platforms and collected from the NCBI GEO. We used only datasets containing at least $\geqslant$20 samples, to ensure that correlations in the coexpression networks were very robust. Each microarray dataset is modeled as a coexpression graph following the method introduced in Sect. 2.1.3.

We identified 4,327 RHSs, each of which contains at least five member genes and occur in at least five networks. The minimum "heaviness" of these patterns is 0.4. The average size is 8.5 genes, and the average recurrence is 10.1 networks. To assess the quality of these RHSs, we evaluate the functional homogeneity of their member genes using both Gene Ontology Analysis and KEGG pathway analysis.

For each RHS, we test its enrichment for specific Gene Ontology (GO) biological process terms and GO cellular component terms [14]. To ensure the specificity of GO terms, we removed from consideration any terms associated with more than 500 genes. If the member genes of a RHS are enriched in a GO term with a hypergeometric p-value less than 0.001, we declare the RHS to be functionally homogeneous. Our results show that 59.7% of RHSs with $\geqslant$5 member genes, $\geqslant$5 recurrences, and $\geqslant$0.4 heaviness were functionally homogenous. To highlight the significance of this result, we generated random patterns with the same size distribution as

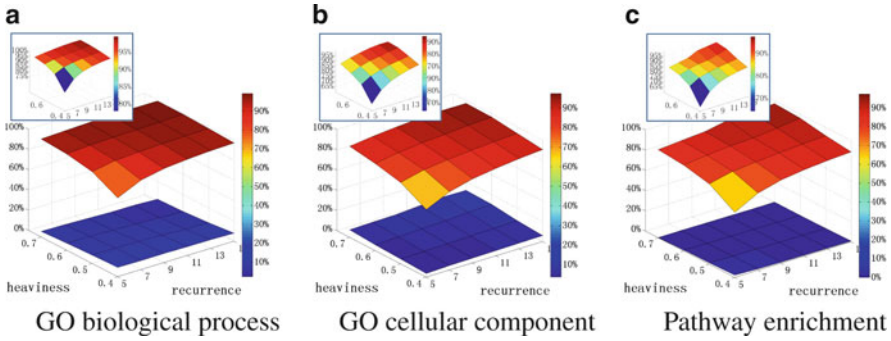


Fig. 6.8 Evaluating the functional homogeneity of RHSs using three forms of enrichment analysis. Each method is presented in two plots: the larger plot shows the difference between enrichment results on RHSs and random patterns; while the smaller plot focuses on the results of RHSs alone. It is obvious that the functional enrichments of RHSs are much greater than those found in random patterns, and also that the quality of the RHSs increases significantly with heaviness and recurrence

the RHSs. Only 9.3% of these patterns were functionally homogenous. The functionally homogenous RHSs cover a wide-range of biological processes, including translational elongation, mitosis, cell cycle, RNA splicing, ribosome biogenesis, histone modification, chromosome localization, spindle checkpoint, posttranscriptional regulation, and protein folding. Our statistical analysis also demonstrates that the greater the heaviness and recurrence, the more likely it is to be functionally homogenous. This relationship is shown in Fig. 6.8a,b.

We used KEGG human pathways[2] to assess the degree to which RHS modules represent known biological pathways. If member genes of a RHS are enriched in a pathway with a hypergeometric p-value less than 0.001, we declare the RHS to be "pathway homogeneous." The results show that 43.5% of RHSs with $\geqslant$5 genes, $\geqslant$5 recurrences, and $\geqslant$0.4 heaviness were pathway homogenous, compared to a rate of 1.7% in randomly generated patterns (Fig. 6.8c). The RHSs are enriched in a variety of pathways: oxidative phosphorylation, cell cycle, cell communication, focal adhesion, ECM-receptor interaction, glycolysis, etc.

5 Conclusion

Biological network data are rapidly accumulating for a wide-range of organisms under various conditions. The integrative analysis of multiple biological networks is a powerful approach to discover meaningful network patterns, including subtle structures and relationships that could not be discovered in a single network.

[2] http://www.genome.jp/kegg/.

In this chapter, we proposed several novel types of recurrent patterns and derived algorithms to discover them. We also demonstrated that the identified patterns can facilitate functional discovery, regulatory network reconstruction, and phenotype characterization. Although we used coexpression networks as examples throughout this work, our methods can be applied to other types of relational graphs for pattern discovery. New challenges will arise as the quantity and complexity of biological network data continue to increase. The wealth of biological data will certainly push the scale and scope of graph-based data mining to the next level.

Acknowledgments The work presented in this chapter was supported by National Institutes of Health Grants R01GM074163, P50HG002790, and U54CA112952 and NSF Grants 0515936, 0747475 and DMS-0705312.

References

1. Acar E, Camtepe SA, Krishnamoorthy M, Yener B (2005) Modeling and multiway analysis of chatroom tensors. In: Proc of IEEE Int. Conf. on Intelligence and Security Informatics, pp 256–268
2. Acar E, Aykut-Bingol C, Bingol H, Bro R, Yener B (2007) Multiway analysis of epilepsy tensors. Bioinformatics 23(13):i10–18
3. Aja-Fernández S, de Luis García R, Tao D, Li X (eds) (2009) Tensors in Image Processing and Computer Vision. Advances in Pattern Recognition, Springer
4. Alter O, Golub GH (2005) Reconstructing the pathways of a cellular system from genome-scale signals by using matrix and tensor computations. Proc Natl Acad Sci USA 102(49):17559–17564
5. Alter O, Brown P, Botstein D (2000) Singular value decomposition for genome-wide expression data processing and modeling. Proc Natl Acad Sci USA 97(18):10101–10106
6. Alter O, Brown P, Botstein D (2003) Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms. Proc Natl Acad Sci USA 100(6):3351–3356
7. Barabasi A, Oltvai Z (2004) Network biology: understanding the cell's functional organization. Nature Reviews Genetics 5(2):101–113
8. Breiman L (2001) Random forests. Machine Learning 45(1):5–32
9. Butte AJ, Chen R (2006) Finding disease-related genomic experiments within an international repository: first steps in translational bioinformatics. AMIA Annual Symposium proceedings pp 106–110
10. Butte AJ, Kohane IS (2006) Creation and implications of a phenome-genome network. Nat Biotechnol 24(1):55–62
11. Cattell RB (1952) The three basic factor-analytic research designs-their interrelations and derivatives. Psychological Bulletin 49:499–452
12. Chung FRK (1997) Spectral Graph Theory. No. 92 in CBMS Regional Conference Series in Mathematics, American Mathematical Society
13. Collette Y, Siarry P (2003) Multiobjective Optimization: Principles and Case Studies. Springer
14. Consortium GO (2006) The gene ontology (go) project in 2006. Nucleic Acids Res 34(Database issue):D322–6
15. Ding C, He X, Zha H (2001) A spectral method to separate disconnected and nearly-disconnected web graph components. In: Proc of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM New York, NY, USA, pp 275–280

16. Edgar R, Domrachev M, Lash AE (2002) Gene expression omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Research 30(1):207–210
17. Faloutsos C, Kolda TG, Sun J (2007) Mining large graphs and streams using matrix and tensor tools. In: Proc. of the ACM SIGMOD International Conference on Management of Data, p 1174
18. Geman S, Geman D (1984) Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6: 721–741
19. Gollub J, Ball CA, Binkley G, Demeter J, Finkelstein DB, Hebert JM, Hernandez-Boussard T, Jin H, Kaloper M, Matese JC, Schroeder M, Brown PO, Botstein D, Sherlock G (2003) The stanford microarray database: data access and quality assessment tools. Nucleic Acids Research 31(1):94–96
20. Grahne G, Zhu J (2003) Efficiently using prefix-trees in mining frequent itemsets. In: FIMI'03 Workshop on Frequent Itemset Mining Implementations
21. Hermjakob H, Montecchi-Palazzi L, Lewington C, Mudali S, Kerrien S, Orchard S, Vingron M, Roechert B, Roepstorff P, Valencia A, Margalit H, Armstrong J, Bairoch A, Cesareni G, Sherman D, Apweiler R (2004) IntAct: an open source molecular interaction database. Nucleic Acids Research 32(Database issue):D452–455
22. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA 79(8):2554–2558
23. Hu H, Yan X, Huang Y, Han J, Zhou XJ (2005) Mining coherent dense subgraphs across massive biological networks for functional discovery. Bioinformatics 21(Suppl 1):i213–221
24. Huang Y, Li H, Hu H, Yan X, Waterman MS, Huang H, Zhou XJ (2007) Systematic discovery of functional modules and context-specific functional annotation of human genome. Bioinformatics 23(13):i222–229
25. Kelley B, Sharan R, Karp R, Sittler T, Root D, Stockwell B, Ideker T (2003) Conserved pathways within bacteria and yeast as revealed by global protein network alignment. Proc Natl Acad Sci USA 100(20):11394–11399
26. Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680
27. Kolda TG, Bader BW, Kenny JP (2005) Higher-order web link analysis using multilinear algebra. In: Proc of IEEE Int. Conf. on Data Mining, pp 242–249
28. Koyutürk M, Grama A, Szpankowski W (2004) An efficient algorithm for detecting frequent subgraphs in biological networks. Bioinformatics 20 Suppl 1:i200–207
29. Koyutürk M, Kim Y, Subramaniam S, Szpankowski W, Grama A (2006) Detecting Conserved Interaction Patterns in Biological Networks. J Comput Biol 13(7):1299–1322
30. Koyutürk M, Kim Y, Topkara U, Subramaniam S, Szpankowski W, Grama A (2006) Pairwise alignment of protein interaction networks. J Comput Biol 13(2):182–199
31. Luxburg U (2007) A tutorial on spectral clustering. Statistics and Computing 17(4):395–416
32. Mahoney M, Maggioni M, Drineas P (2008) Tensor-CUR decompositions for tensor-based data. SIAM Journal on Matrix Analysis and Applications 30:957–987
33. Mehan MR, Nunez-Iglesias J, Kalakrishnan M, Waterman MS, Zhou XJ (2009) An integrative network approach to map the transcriptome to the phenome. J Comput Biol 16(8):1023–1034
34. Mitchell JA, Aronson AR, Mork JG, Folk LC, Humphrey SM, Ward JM (2003) Gene indexing: characterization and analysis of nlm's generifs. AMIA Annual Symposium proceedings pp 460–4
35. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of Turán. Canad J Math 17(4):533–540
36. Newman MEJ (2004) Analysis of weighted networks. Phys Rev E 70(5):056131
37. Ng A, Jordan M, Weiss Y (2001) On spectral clustering: Analysis and an algorithm. In: Proc. Advances in Neural Information Processing Systems, pp 849–856
38. Omberg L, Golub GH, Alter O (2007) A tensor higher-order singular value decomposition for integrative analysis of DNA microarray data from different studies. Proc Natl Acad Sci USA 104(47):18371–18376

39. Papadimitriou CH (1981) On the complexity of integer programming. Journal of the ACM 28(4):765–768
40. Papin J, Price N, Wiback S, Fell D, Palsson B (2003) Metabolic pathways in the post-genome era. Trends Biochem Sci 28(5):250–258
41. Serrano MA, Boguñá M, Vespignani A (2009) Extracting the multiscale backbone of complex weighted networks. Proc Natl Acad Sci USA 106(16):6483–6488
42. Sharan R, Suthram S, Kelley RM, Kuhn T, McCuine S, Uetz P, Sittler T, Karp RM, Ideker T (2005) Conserved patterns of protein interaction in multiple species. Proc Natl Acad Sci USA 102(6):1974–1979
43. Smilde A, Bro R, Geladi P (2004) Multi-way Analysis: Applications in the Chemical Sciences. Wiley, West Sussex, England
44. Suman B, Kumar P (2006) A survey of simulated annealing as a tool for single and multiobjective optimization. Journal of the Operational Research Society 57(10):1143–1160
45. Sun J, Tao D, Faloutsos C (2006) Beyond streams and graphs: dynamic tensor analysis. In: Proc of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 374–383
46. Sun J, Tao D, Papadimitriou S, Yu PS, Faloutsos C (2008) Incremental tensor analysis: Theory and applications. ACM Transactions on Knowledge Discovery from Data 2(3)
47. Sun J, Tsourakakis C, Hoke E, Faloutsos C, Eliassi-Rad T (2008) Two heads better than one: pattern discovery in time-evolving multi-aspect data. Data Mining and Knowledge Discovery 17(1):111–128
48. Tao D, Song M, Li X, Shen J, Sun J, Wu X, Faloutsos C, Maybank SJ (2008) Bayesian tensor approach for 3-d face modeling. IEEE Trans Circuits Syst Video Techn 18(10):1397–1410
49. Tucker LR (1966) Some mathematical notes on three-mode factor analysis. Psychometrika 31:279–311
50. Wu LF, Hughes TR, Davierwala AP, Robinson MD, Stoughton R, Altschuler SJ (2002) Large-scale prediction of saccharomyces cerevisiae gene function using overlapping transcriptional clusters. Nature Genetics 31(3):255–265
51. Yan X, Mehan MR, Huang Y, Waterman MS, Yu PS, Zhou XJ (2007) A graph-based approach to systematically reconstruct human transcriptional regulatory modules. Bioinformatics 23(13):i577–586
52. Zhang T (2008) Multi-stage convex relaxation for learning with sparse regularization. In: Proc. of Advances in Neural Information Processing Systems, pp 1929–1936
53. Zhang T (2009) Multi-stage convex relaxation for non-convex optimization. Tech. rep., Rutgers University
54. Zhou X, Kao MJ, Wong WH (2002) Transitive functional annotation by shortest-path analysis of gene expression data. Proc Natl Acad Sci USA 99(20):12,783–12,788
55. Zhou X, Kao M, Huang H, Wong A, Nunez-Iglesias J, Primig M, Aparicio O, Finch C, Morgan T, Wong W, et al (2005) Functional annotation and network reconstruction through cross-platform integration of microarray data. Nature Biotechnology 23:238–243