New Algorithms for Mining Network Datasets: Applications to Phenotype and Pathway Modeling

Ying Jin

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Naren Ramakrishnan, Chair
Edward A. Fox
Lenwood S. Heath
Richard F. Helm
T.M. Murali

December 8, 2009
Blacksburg, Virginia

Keywords: Biological networks, relative importance methods,
graph minimal separators, biclusters, partial orders.

New Algorithms for Mining Network Datasets: Applications to Phenotype and Pathway Modeling

Ying Jin

(ABSTRACT)

Biological network data is plentiful with practically every experimental methodology giving 'network views' into cellular function and behavior. Bioinformatic screens that yield network data include, for example, genome-wide deletion screens, protein-protein interaction assays, RNA interference experiments, and methods to probe metabolic pathways. Efficient and comprehensive computational approaches are required to model these screens and gain insight into the nature of biological networks. This thesis presents three new algorithms to model and mine network datasets. First, we present an algorithm that models genome-wide perturbation screens by deriving relations between phenotypes and subsequently using these relations in a local manner to derive gene-phenotype relationships. We show how this algorithm outperforms all previously described algorithms for gene-phenotype modeling. We also present theoretical insight into the convergence and accuracy properties of this approach. Second, we define a new data mining problem—*constrained minimal separator mining*—and propose algorithms as well as applications to modeling gene perturbation screens by viewing the perturbed genes as a *graph separator*. Both of these data mining applications are evaluated on network datasets from *S. cerevisiae* and *C. elegans*. Finally, we present an approach to model the relationship between metabolic pathways and operon structure in prokaryotic genomes. In this approach, we present a new pattern class—biclusters over domains with supplied partial orders—and present algorithms for systematically detecting such biclusters. Together, our data mining algorithms provide a comprehensive arsenal of techniques for modeling gene perturbation screens and metabolic pathways.

This work is supported by NSF ITR Grant (#0428344 - "Computational Models for Gene Silencing: Elucidating a Pervasive Biological Defense").

Acknowledgments

First I offer my sincerest gratitude to my supervisor, Dr. Naren Ramakrishnan, for his invaluable advise, patience, encouragement throughout my Ph.D. research. I attribute the level of my Ph.D. degree to his encouragement and effort and without him this thesis, too, would not have been completed or written.

I thank Dr. Heath for his invaluable guidance and suggestions on the papers and his support through CMGS project.

As committee members, Dr. Heath, Dr. Fox, Dr. Helm, and Dr. Murali provided their precious suggestions and encouragement during each conversation we have had. Their unique views to the area I have been working on have revealed interesting issues and motivated enhancement to my research work.

Many thanks to Amrita Pati and Karsten Klage for their support on my research. I thank every one in softlab. In my daily work, I have been blessed with friendly and cheerful lab-mates. Meanwhile I appreciate the graceful help from all my friends that cheered my life.

I thank my brother Jinyuan and my parents for their support, patience, and love, especially for taking care of our lovely son. Finally, I thank my husband David for his support, inspiration, encouragement, and discussing on my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last two decades, data mining methodologies have become extremely popular in many scientific domains, including bioinformatics [85]. A key factor has been the availability of high-throughput biotechnologies (e.g., microarrays, proteomic assays) for studying complex biological processes. Many of the datasets obtained through these methodologies yield 'network views' of biology, i.e., the data can be viewed as capturing information about different biological networks.

We categorize these networks into two groups: *undirected* and *directed* networks.

1. **Undirected Networks:** Three key data sources that yield undirected network information are protein-protein interaction experiments, gene co-expression data, and genome-wide deletion mutants.

   For instance, proteins form cellular complexes (e.g., by binding to each other) to perform structural and functional roles. Some proteins might interact together to form a channel that allows substrate to move from one enzyme to the next without diffusional losses away from the biosynthetic assembly. Such associations between proteins are referred to as protein-protein interactions (PPI). Besides being objects of intrinsic interest, protein complexes have been used to support phenotype predictions [22]. From a computational perspective, protein-protein interactions are usually modeled as un-weighted, undirected networks where the nodes are proteins.

   Gene expression datasets being generated from transcriptional profiling provide another source of undirected network data. Here, the nodes are genes and the edges (possibly) represent co-regulation of the associated genes. The weight of the edge typically indicates the strength of the co-expression. Gene co-expression data also has been used to elucidate relationships between genes and phenotypes [56, 61].

   Technologies such as genome-wide deletion mutants [64, 76], using either homologous recombination (knockouts) or RNA interference (knockdowns) [13, 21, 36], are a third source of data that more directly help associate phenotypes to thousands of genes. Here, gene-phenotype relations can be modeled as a bipartite network between genes and phenotypes.

The link between a gene and a phenotype means that perturbing the gene gives rise to the corresponding phenotype.

2. **Directed Networks:** Directed networks are also popular in bioinformatics databases, with sources being transcriptional regulation, biochemical pathways, and signal transduction networks. Sources of data for directed networks are based upon transcriptional regulation experiments, biochemical pathway analysis, and signal transduction network perturbations.

   A transcription factor (a type of protein) can either activate or inhibit the expression of a gene. A transcriptional regulatory network is a collection of genes coding for transcription factors and their regulated targets. The hierarchical structures of the transcriptional network of many model organisms have been studied [42, 87]. Such a network is best modeled as a directed graph, where the direction of an edge distinguishes the regulator from the regulated.

   A biochemical pathway can be viewed as a series of chemical reactions, where downstream reactions process the products of the upstream reactions. A metabolic network is a collection of such biochemical pathways. The structure of a metabolic network is more complicated than the previously described networks. It can potentially embody multiple types of relations (e.g., between reactions and reactions, and between enzymes and reactions). There are some open databases that integrate enzymes/reactions/pathways data of multiple organisms [4, 48, 49].

   Signaling transduction networks comprise the chemical processes by which cells detect and respond to changes in the environment. Once a receptor receives a signal, it transmits the information into the cell resulting the response. Similar to a metabolic network, signal transduction networks also involve multiple relations over many different domains.

In this thesis, we leverage diverse network sources to yield insight into two key problems: phenotype and pathway modeling. We begin with motivations for both of them. A "principal goal of [modern] genetic research is to identify specific genotypes that are associated with human phenotypes" [23]. One of the most compelling areas of interest is understanding the nature of human diseases and discovering new drug targets for them based on their genetic causes [11]. However, many disease-causing genes are still unknown. Moreover, it is infeasible to experimentally test genes individually for the purposes of discovering drug targets. Hence it is necessary to use computational approaches to identify candidate genes for controlling a phenotype by utilizing all available data sources. Extending this argument further, the functional state of a metabolic network can be considered as a phenotype. Successfully modeling the relations between gene regulation and metabolic pathways is very important to understanding any causality relationships between genes and phenotypes. Although many existing data mining approaches work for biological data, new and effective algorithms are still required considering the speciality of these applications and the characteristics of data. In this dissertation, we develop new methodologies for mining network data sets and apply them to modeling phenotypes and metabolic pathways.

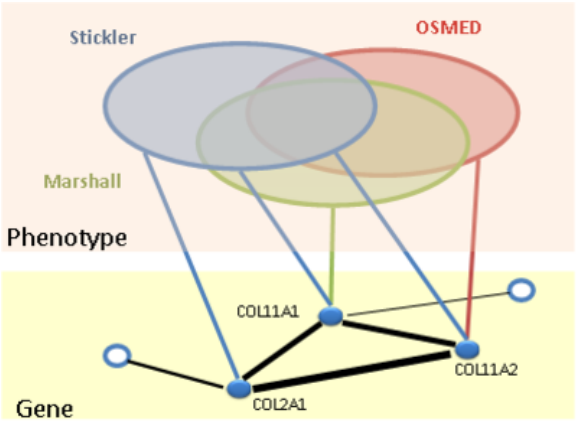


Figure 1.1: An example of relations between genes and phenotypes. The thickness of the edge represents the strength of the connections.

1.1 Research Questions

We propose the following research questions involving mining undirected and directed network datasets. Each of these questions arises from a biologically motivated context involving the modeling of phenotypes and/or metabolic pathways.

1. **How can we predict unknown relationships between genes and phenotypes using data from protein-protein interactions and gene perturbation screens?**

   It is clear that gene perturbation screens yield direct insight into the relationship between genes and phenotypes. It also has been shown that genes encoding for proteins in the same protein complex are likely to exhibit the same phenotypes when perturbed [22]. Therefore, knowledge about protein-protein interactions is a second basis for understanding phenotypes.

   Previous studies have shown that phenotypically similar diseases are often caused by functionally related genes [65, 66]. For example, Stickler, Marshall, and OSMED syndromes are caused by mutations in functionally related genes—COL2A1, COL11A1, and COL11A2—all of which code for collagens [2, 31, 79]. Fig. 1.1 depicts relations between collagens and the associated phenotypes. These syndromes share similar phenotypes, and the genes causing these syndromes are functionally related. Hence, two similar phenotypes are likely to share the same causative genes. This observation tells us that the relationship between phenotypes is probably important to model, in order to better predict relations between genes and phenotypes.

   In Chapter 2, we present a new method (P^3) [44] that first derives *relations between phenotypes* and uses them in a local manner to derive new gene-phenotype relationships. This work has parallels to the *user-based* [39] and *item-based* [75] approaches from the recommender systems literature. We show how P^3 consistently outperforms *all* other methods for phenotype modeling.

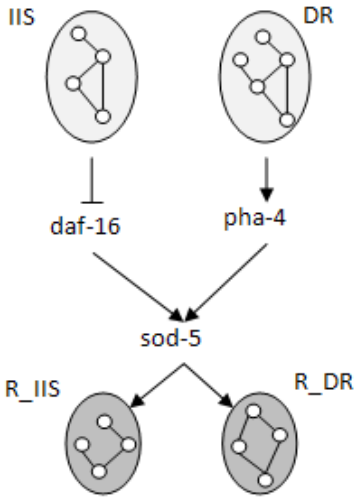


Figure 1.2: Reduced IIS and dietary restriction longevity.

2. **How can we identify a minimal set of genes to knockout/remove from a network such that certain functions are blocked but with minimum side effects to other functions?**

   Genes are situated as critical players in complex biological networks. For instance, some genes encode for enzymes that serve to catalyze crucial biochemical pathways. We can study the effect of knocking out/removing a gene on the overall structure of a network.

   One example arises in the study of desiccation tolerance in *C.elegans*. Figure 1.2 shows a brief description of the important pathways and genes/proteins in this process. It is known that the insulin/IGF-1 signalling (IIS) pathway is a key regulator of the aging process in *C.elegans*. Reduced IIS increases longevity by activating the function of *daf-16*. Further, it is known that *pha-4* regulates diet-restriction-mediated longevity. *pha-4* is not required for the response of reduced IIS, and *daf-16* is not required in the regulation of longevity under dietary restriction. However, both *daf-16* and *pha-4* positively regulate a superoxide-dismutase (*sod-5*), that slows down metabolism. We can thus view these two independent pathways as being connected through a critical point (*sod-5*). Biologists will select *sod-5* to mutate in order to investigate the survival situations under desiccation stress with normal metabolism, instead of *pha-4* or *daf-16*. This idea naturally maps to target selection based on the topological location of *sod-5* in the pathway network as shown in Figure 1.2.

   We consider the set of perturbed genes as a *graph separator* and aim to understand which parts of the underlying network are disconnected as a result of removing the set. In Chapter 3, we define a new data mining problem—*constrained minimal separator mining*—and propose algorithms as well as applications to modeling gene perturbation scree [45].

3. **How can we model the relationship between transcriptional regulatory networks and metabolic networks?**

   Transcriptional regulatory relationships and the structure of metabolic networks are highly

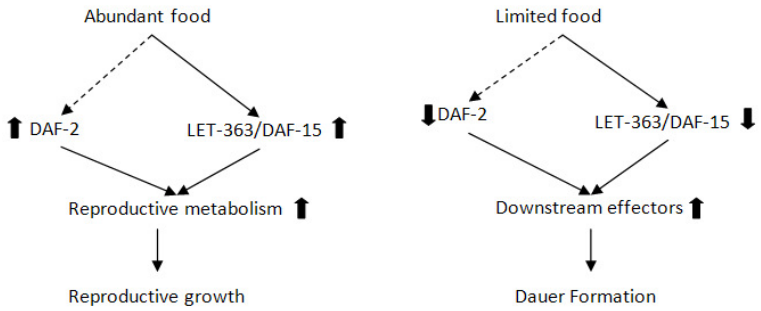


Figure 1.3: *C. elegans* larval development, metabolism, and longevity. (left) Sufficient nutrient condition. (right) Insufficient nutrient condition.

inter-twined. For example, a partial model for regulation of *C. elegans* larval development, metabolism, and longevity is shown in Figure 1.3. Both *daf-15* and *daf-2* pathways are essential for larval development. Figure 1.3 (left) shows the case when food is abundant. *let-363*/*daf-15* is up-regulated and in turn transduces a sufficient nutrient signal to permit growth to the reproductive adult. This signal is also required for *daf-2*/insulin signaling to stimulate growth. When nutrients are limited as shown in Figure 1.3, the *let-363*/*daf-15* signal is insufficient, with concomitant down regulation of the *daf-2* pathway, and the organism will enter the dauer stage.

To model the close relationships between gene regulation and pathway regulation, we exploit natural orderings between reactions and genes. There is a partial order that can be posed over reactions, since the products of one reaction are the reactants to the subsequent reaction. Similarly, there is a partial order between genes that reflects their location and hence possible co-regulation. We formulate a data mining problem that aims to identify concerted clusters of genes and reactions that obey/reflect these partial orders. In this problem, we generalize the notion of biclusters over sets to biclusters over partially ordered sets in Chapter 4. We present algorithms for mining such biclusters on the relations between reactions and co-regulated genes. We use a prokaryotic example (*E.coli*) for ready availability of data.

1.2 Outline

The remainder of this dissertation is organized on the base of the research problems. Related work is presented contextually, within the setting of each chapter.

- Chapter 2 describes a relative importance based method for modeling gene perturbation screens.

- Chapter 3 describes algorithms for mining constrained graph minimal separators and applications in the study of multiple knockouts/knockdowns.
- Chapter 4 describes algorithms for computing biclusters over domains with supplied partial orders, and presents applications to modeling pathways and transcriptional regulatory relationships.
- Chapter 5 concludes this dissertation and outlines possible directions for future work.

Chapter 2

Relative Importance Methods to Model Gene Perturbation Screens

With the advent of high-throughput gene perturbation screens (e.g., RNAi assays, genome-wide deletion mutants), modeling the complex relationship between genes and phenotypes has become a paramount problem. One broad class of methods uses 'guilt by association' to impute phenotypes to genes based on the interactions between the given gene and other genes with known phenotypes. Yet these methods are inadequate for genes that have no cataloged interactions but which nevertheless are known to result in important phenotypes.

In this chapter, we present an approach to first model relationships between phenotypes using the notion of 'relative importance' and subsequently use these derived relationships to make phenotype predictions. This new approach sheds insight into relations between phenotypes and yields improved accuracy on *S. cerevisiae* deletion mutants and *C. elegans* knock-down datasets.

2.1 Introduction

There are now a variety of mechanisms to study loss of function phenotypes in specific cell types or at different stages of development in an organism. Genome wide deletion mutants, e.g., for *Saccharomyces cerevisiae* [64, 76], use homologous recombination to replace genes with targeted cassettes so that the resulting strain can be screened for specific phenotypes. RNA interference methodologies, in organisms such as *Caenorhabditis elegans* [72,80], use post-transcriptional gene silencing to degrade specific RNA molecules, thus causing a drastic attenuation of gene expression. Since RNAi may not completely deplete the expressed RNA molecules, its use is referred to as a 'knockdown', in contrast to a complete 'knockout' exhibited by a deletion mutant. Through the use of high-throughput screens, both these techniques now support large scale phenotypical studies.

A central goal of bioinformatics research is to model the phenotypic effects of gene perturbations.

The mapping between gene function and expressed phenotype is complex. A single gene perturbation (through deletion or RNAi interference) can lead to a cascade of changes in transcription or post-transcriptional pathways. It is impractical to make a comprehensive empirical analysis when there is a large number of candidate genes. An emerging area of interest therefore is to use diverse, highly heterogeneous, data (e.g., microarrays, RNAi studies, protein-protein interaction assays) to computationally model phenotype effects for mutations.

Previous studies have shown that by considering interactions between candidate genes and target genes (which have been known to result in a desired phenotype) the accuracy of phenotype prediction can be improved. Examples of interactions that have been considered include physical interactions between proteins [67], interactions underlying protein complexes [53], and integrated gene networks constructed from multiple data sources [56]. Most of these methods can be classified as 'direct' methods since they require a direct interaction between a gene and another gene with the target phenotype in order to predict the phenotype for the given gene.

The combination of gene expression and protein interaction data have been used to to prioritize genes, e.g., CGI [57] and GeneRank [61]. In addition to direct interactions, these methods take into account indirect interactions, i.e., links from genes to target genes through other intermediate genes. However, these approaches assume that there is at least one path from a candidate gene to some target gene(s). Since many genes do not have any catalogued interactions, this limits the applicability of these approaches.

Markowetz *et al.* [59] proposed the NEM (nested effects models) approach to rank genes according to subset relations between phenotypes. NEM uses phenotype profiles only, i.e., it does not consider any protein-protein interactions. While this overcomes the limitations mentioned previously, NEM has shortcomings in scalability with respect to the number of phenotypes. To overcome the increased computational cost, NEM focuses only on inference from pairwise and triple relations.

The key contributions of this chapter are several fold. We propose a new graph theoretic approach to predicting phenotype effects of gene perturbation using phenotype relations (P^3). Our approach focuses on relative importance methods to infer relations between phenotypes and uses these relations to predict phenotypic effects. We integrate phenotype profiles with the gene network to derive phenotype relations. It is assumed that genes that are tightly connected are likely to share the same phenotypes. We use a weighted directed graph to model the relations between phenotypes such that more complicated relations can be illustrated and interpreted instead of just subset relations. Since predictions are carried out purely based on the phenotype relations derived, there is no requirement for known interaction paths from candidate genes to target genes. Furthermore, once the relations between phenotypes are derived, they can be used repetitively in the prediction process. In particular, complete perturbation effects across all phenotypes can be predicted simultaneously from the relations between known phenotypes and others. Therefore, P^3 is more effective for large-scale phenotype prediction than previous methods that rank genes for each phenotype, one at a time. Experimental results on *S. cerevisiae* and *C. elegans* networks also show that our approach outperforms the direct and GeneRank methods consistently. In particular, for genes without any interactions in *S. cerevisiae*, we show that our method can predict 96% of their

phenotypes with AUC (area under ROC curve) greater than 0.8, and 60% of the phenotypes in *C. elegans*.

We also aim to explain the success of our framework using theoretical concepts as well as further experimental exploration. First, we develop a hybrid importance weighting method that interpolates between relative and global importance methods and demonstrate that relative importance is crucial to the success of P^3. Second, we provide a new semi-definite programming formulation for optimizing the initial settings of the relative importance algorithm and discuss its implications. Finally, we provide a formal proof of the convergence of the relative importance algorithm used in P^3.

2.2 Problem Definition

Let $\mathcal{G} = \{g_1, g_2, ..., g_n\}$ be a set of genes. Genes interact with each other, and these interactions are encoded in an $n \times n$ connection matrix W, where $w_{i,j}$ denotes the weight of the connection between gene g_i and gene g_j. W is a symmetric matrix whose diagonal is uniformly 0. Let $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ be a set of phenotypes. Each phenotype has a set of genes associated with it, represented by a vector $\vec{p_j} = \langle v_1, v_2, ..., v_n \rangle$, where $v_i = 1$ indicates that gene g_i is known to result in p_j. If the relationship between the gene g_i and the phenotype p_j is unknown, then $v_i = 0$. These vectors are grouped together to form an $m \times n$ phenotype-gene matrix V, where rows are phenotypes and columns are genes.

The objective is to rank the set of genes to each phenotype to find unknown gene-phenotype relations. We hypothesize that similar phenotypes are associated with closely functionally related genes. The following example shows the importance of taking into account both gene interactions and phenotype relations.

Table 2.1 describes an example of phenotype profiles resulting from many gene perturbations. These phenotype profiles constitute one part of the training data for phenotype prediction. Table 2.2 shows the test data for evaluating prediction performance. In the two tables, each row represents a phenotype and each column a gene. The cell value indicates whether the gene perturbation exhibits the corresponding phenotype, e.g., perturbing g_1 gives rise to p_1 but not p_2 and p_3. Gene g_7 is known to give rise to phenotype p_3 (see the last column of Table 2.1), and we seek to determine if g_7 also will cause p_2 and/or p_1.

A second form of data available is a gene network as shown in Figure 2.1 (left), that shows interactions between genes. For ease of interpretation, genes that result in the same phenotype are grouped in Figure 2.1 (left).

- **Using phenotype profiles:** If we were to use only Table 2.1 to make a prediction, it is not clear whether g_7 should result in p_1 or p_2. p_1 and p_2 involve three genes each, and p_3 has (exactly) one gene in common with both sets. Obviously, p_1 and p_2 have an equal chance to be predicted, no matter what association measure is used.

Table 2.1: Input data for phenotype prediction.

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
p_1	1	1	0	0	0	1	?
p_2	0	0	1	1	1	0	?
p_3	0	0	0	1	0	1	1

Table 2.2: Test data.

	g_7
p_1	0
p_2	1
p_3	1

- **Using network information:** If we assume that all links in Figure 2.1 (left) have the same weight, then in fact the prediction result will be p_1. To see this, observe that g_7 has only one interaction partner g_2, and it is known that g_2 contributes to p_1 only. And there are no paths from g_7 to any genes resulting in phenotype p_2. Hence, no matter what graph theoretic methods are used, p_1 has a better chance of being predicted.

As we can see, neither the phenotype profiles or network information allows us to accurately predict the phenotypes as shown in the test data (Table 2.2). We propose to combine the selective superiorities of the two methods to model phenotypes. In this process, we develop a method that resembles a collaborative filtering algorithm [38] used in recommender systems research. First, we derive relationships between phenotypes from Table 2.1 and Figure 2.1 (left). Figure 2.1 (right) demonstrates the relationships between phenotypes obtained by applying our algorithm presented in the following section. The value on the arrow from phenotype p_i to phenotype p_j denotes the tendency that a gene perturbation causing p_i also causes p_j. From such a relation, we can predict that if a gene perturbation results in p_3, then it is more likely to result in p_2 rather than p_1. Some characteristics of existing methods and our approach are listed in Table 2.3.

2.3 Methods

2.3.1 Inferring Relations Between Phenotypes

As stated earlier, inferring relations between phenotypes is a one-time cost and can be amortized over the prediction process. Our method is motivated by the study of relative importance in networks [86]. Original link analysis methods, such as PageRank [68] and HITS [51], rank nodes according to their "importance" or "authority" in the entire network. Relative importance analysis focuses on determining the importance of nodes with respect to a subset of the network, called the "root set." Multiple algorithms have been proposed for relative importance computation, such as

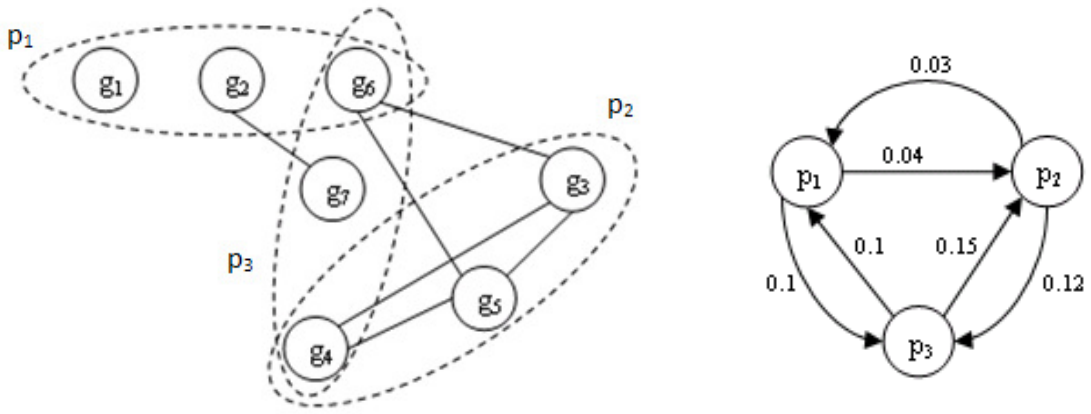


Figure 2.1: (left) Example of gene network. (right) Induced relationships between phenotypes.

Table 2.3: Comparison of P^3 to other methods for phenotype prediction.

Method	Use phenotype profiles?	Use gene interactions?	Ability to rank phenotypes?	Induce phenotype relations?
Wormnet (Direct)		√		
GeneRank		√		
NEM	√		√	√
P^3	√	√	√	√

k-short path, k-short node-disjoint paths, PageRank with priors, HITS with priors, and the k-step Markov approach, which are all surveyed by White and Smyth [86].

Using the same notation defined in the problem definition, suppose that there are n genes $G = \{g_i \mid 1 \leq i \leq n\}$, and m phenotypes $P = \{p_i \mid 1 \leq i \leq m\}$ in a study. Let W denote the connection matrix of the network between genes, and V indicate the relations between phenotypes and genes. Given a phenotype p_j, genes resulting in this phenotype form a root set R_j. Similar to PageRank with priors, each gene is assigned a prior rank score, as shown in Equation 2.1. Observe that the sum of all initial rank scores is 1. If $R_j \neq 0$,

$$r^0_{g_i p_j} = \begin{cases} \frac{1}{\|R_j\|} & \text{if } g_i \in R_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Let $N(g_i) = \{g_j \mid w_{i,j} > 0, i \neq j, \text{and } g_j \in G\}$ denote the set of all other genes that interact with g_i. Define parameter β, $0 \leq \beta \leq 1$, to be the relative weight between the original score of a gene and the score that results through the influence of its neighbors. The formula for iteratively computing gene rank scores is shown in Equation 2.2.

$$r_{g_i p_j}^{k+1} = \beta r_{g_i p_j}^{0} + (1-\beta)\left(\sum_{g_l \in N(g_i)} \frac{w_{i,j}}{\pi_{g_i}} r_{g_j p_j}^{k}\right). \tag{2.2}$$

Here, $\pi_{g_i} = \sum_{j=1}^{n} w_{i,j}$ is the total weight of interactions involving gene g_i. k in the equation indicates the number of iterations. After convergence, we obtain rank scores of all genes with respect to each phenotype. The convergence of this algorithm is discussed in Section 2.5. The above procedure can be repeated for every phenotype to obtain the corresponding list of rank scores of all genes. The list of rank scores of genes to a phenotype corresponds to a vector $\vec{\mathcal{R}_{p_i}} =< r_{g_1}, ..., r_{g_n} >$, where r_{g_k} is the rank score of g_k. Let C denote a $m \times m$ "closeness" matrix of phenotypes, where both rows and columns are phenotypes, and each entry $c_{i,j}$ stores the closeness value from phenotype p_j to p_i. The previous example shown in Figure 2.1 (right) is a network view of the closeness matrix between phenotypes. It is defined as the p_i's average rank scores of genes causing p_j. The formula is given in Equation 2.3, where $\vec{p_j}^T$ is the transpose of $\vec{p_j}$.

$$c_{i,j} = \begin{cases} \vec{p_j}^T \times \vec{\mathcal{R}_{p_i}} & i \neq j, \\ 1 & i = j. \end{cases} \tag{2.3}$$

Note that the closeness matrix C is not necessarily symmetric, since the rank score of a gene to a phenotype depends on the scores of its neighbors, but for two phenotypes p_i and p_j, genes involved in phenotype p_i may not have the same neighbors as genes involved in phenotype p_j. For simplicity, the diagonal of the matrix is set to 0, because the closeness of a phenotype to itself is not of interest. This matrix thus maps to a weighted directed graph, such as seen in Figure 2.1 (right), where nodes are phenotypes, and the weight of the directed edge from phenotype p_i to phenotype p_j is $c_{i,j}$. After the whole matrix C is computed, prediction is carried out using this matrix.

2.3.2 Predicting Phenotype Effects of Gene Perturbations

Algorithms for ranking genes to a phenotype and ranking phenotypes for a gene using the phenotype graph are described below.

Ranking Genes for a Phenotype

Given a phenotype p, suppose that there is a gene g which is known to result in phenotypes $\{q_1, ...q_k\}$. The closeness of phenotype q_i, $1 \leq i \leq k$, to p is the weight of the edge from p to q_i in the phenotype graph. There are multiple ways to define the rank score of a gene g to the phenotype p, for example, we can utilize the maximum closeness from q_i, $1 \leq i \leq k$, to p. Here, we used the average closeness from known phenotypes of the gene to the target phenotype. The rank scores of all genes to all target phenotypes can be calculated simultaneously by a simple

matrix computation, as shown in Equation 2.4.

$$R_G = V' \times C \tag{2.4}$$

V', with entries $v'_{i,j} = \frac{v_{j,i}}{\sum_{k=1}^{m} v_{k,i}}$, is obtained by transposing the phenotype-gene matrix V and dividing each entry by the number of 1s in the corresponding row. R_G is thus an $n \times m$ matrix, where rows are genes and columns are phenotypes, and the value of each cell is the rank score of the gene to the corresponding phenotype.

The matrices V', C, and R_G for the example shown in Figure 2.1 are demonstrated in Figure 2.2. Each column of the matrix shown in Figure 2.2(c) represents the list of ranking values of genes for the corresponding phenotype.

	p_1	p_2	p_3
g_1:	1	0	0
g_2:	1	0	0
g_3:	0	1	0
g_4:	0	0.5	0.5
g_5:	0	1	0
g_6:	0.5	0	0.5
g_7:	0	0	1

(a)

	p_1	p_2	p_3
p_1:	1	0.04	0.1
p_2:	0.03	1	0.12
p_3:	0.1	0.15	1

(b)

	p_1	p_2	p_3
g_1:	1	0.04	0.1
g_2:	1	0.04	0.1
g_3:	0.03	1	0.12
g_4:	0.065	0.575	0.56
g_5:	0.03	1	0.12
g_6:	0.55	0.095	0.55
g_7:	0.1	0.15	1

(c)

Figure 2.2: (a) V', (b) C, and (c) R_G for the example shown in Figure 2.1.

Ranking Phenotypes for a Gene

Given a gene g, assume that it is known to result in phenotypes $\{p_1, ..., p_k\}$. For any other phenotype p' in the phenotype graph, the closeness from p' to phenotype p_i, $1 \leq i \leq k$ is the weight of the edge from p_i to p'. The method of ranking phenotypes to a gene is very similar to ranking genes for a phenotype, described above. In ranking genes, the weights on the edges incident on phenotypes $\{p_1, ..., p_k\}$ are used, but in ranking phenotypes, the edges outgoing from phenotypes $\{p_1, .., p_k\}$ are considered. The rank score of phenotype p to gene g is the average of the closeness values from p' to phenotypes $\{p_1, ..., p_k\}$. Analogously as stated earlier, rank scores of all phenotypes to all genes can be computed at the same time. Equation 2.5 describes the method, where R_P is the resulting rank score matrix.

$$R_P = V' \times C^T \tag{2.5}$$

The only difference from the method to ranking genes is that the transpose of the closeness matrix is used here.

The matrices V', C^T, and R_P for the example shown in Figure 2.1 are demonstrated in Figure 2.3. Each row of the matrix R_P shown in Figure 2.3(c) represents the list of ranking values of phenotypes for the corresponding gene.

	p_1	p_2	p_3
g_1:	1	0	0
g_2:	1	0	0
g_3:	0	1	0
g_4:	0	0.5	0.5
g_5:	0	1	0
g_6:	0.5	0	0.5
g_7:	0	0	1

(a)

	p_1	p_2	p_3
p_1:	1	0.03	0.1
p_2:	0.04	1	0.15
p_3:	0.1	0.12	1

(b)

	p_1	p_2	p_3
g_1:	1	0.03	0.1
g_2:	1	0.03	0.1
g_3:	0.04	1	0.15
g_4:	0.07	0.56	0.575
g_5:	0.04	1	0.15
g_6:	0.55	0.075	0.55
g_7:	0.1	0.12	1

(c)

Figure 2.3: (a) V', (b) C^T, and (c) R_P for the example shown in Figure 2.1.

2.3.3 Computational Complexity

We analyze the computational complexity of the algorithm for each step separately. The second step computes ranks for all genes to all phenotypes simultaneously by a simple matrix multiplication. The time complexity is $O(n \times m^2)$, where n is the number of genes and m is the number of phenotypes. The most time consuming part of the algorithm is the first step. Here, the time complexity is $O(K \times (2|E|))$, the same as the standard PageRank algorithm, where K is the number of iterations and E is the number of edges in the network. Since $\mathcal{N}$ is an undirected graph, in each iteration each edge is accessed two times. It has been theoretically proven that the PageRank algorithm will converge [20], and it has been established experimentally that the number of iteration is much less than the number of nodes in the network, i.e., $I \ll n$ [68]. Many optimization mechanisms have been proposed to improve the computational performance of PageRank [37,46,47,55]. All of them are applicable to the approaches proposed in this chapter.

2.4 Experimental Results

We illustrate the effectiveness of our methodology by comparing it to the "Direct" method (as used in Lee et al. [56]) and GeneRank [61] on two real datasets: deletion mutants on yeast and an RNAi study of early embryogenesis in the *C. elegans* nematode. We further analyze the phenotype graphs derived by clustering phenotypes with high closeness values and present a biological interpretation.

2.4.1 Data

Two datasets are used in this study: the dataset of *C. elegans* RNAi induced early embryo defects [80] and the yeast knockout dataset from the Munich Information Center for Protein sequences (MIPS) database [60].

We focus on 45 RNAi induced defect categories in the *C.elegans* early embryo (data available

in [70]) and use an interaction network extracted from Wormnet [56]. The original core Wormnet contains 113,829 connections and 12,357 genes. To compare with the Direct and GeneRank methods, we select genes resulting in at least two early embryo defects and interacting with at least one other gene, and retain all interactions between them in Wormnet. To evaluate the applicability of P^3 on predicting phenotypes for genes without interactions, we prepare another gene set that retains genes without any interactions.

In the yeast data, the underlying network involves protein-protein interactions and is built by combining the yeast protein interaction data from several sources (CYGD [34], SGD [15], and BioGrid [12]). Phenotypes and genes are selected according to the same criteria as above. The statistics of these datasets are listed in Table 4.1.

2.4.2 Experiment Setup

We implement the Direct method and use the log-likelihood value of each interaction published with Wormnet as the edge weights for the *C. elegans* network. For a given phenotype, genes known to result in that phenotype are considered as the root set. The rank score of other genes are the sum of the log-likelihoods of interactions with the root set. In the case of yeast, we simply set the same weight on all interactions.

Table 2.4: Statistics of datasets used in this work.

Organism	Genes	Interactions	Phenotypes
Caenorhabditis elegans	420	6677	45
Saccharomyces cerevisiae	1232	13228	72

In addition to the connectivity matrix of the network, GeneRank has another input, namely the expression changes vector, which is used to set initial ranks. In our case, we use a binary phenotype signature vector, where 1 means that the corresponding gene is known to show that phenotype, 0 otherwise. There is also a parameter d that determines relative weights of expression changes and connectivity information to the rank value. We tried multiple values on d from 0.1 to 0.9 with interval 0.1, and chose the one that gives optimal prediction results in performance comparison (0.1). The implementation published with the original paper is used.

To compare with the above methods, our algorithm P^3 for ranking genes for a given phenotype is applied. Another algorithm to ranking phenotypes for a given gene is used to predict phenotypes for genes without any interactions. There is one parameter β in P^3 to derive relations between phenotypes. We studied different values on β from 0.1 to 0.9 with step 0.1, and found that 0.6 gives the best performance. We used 0.6 in all the experiments described below.

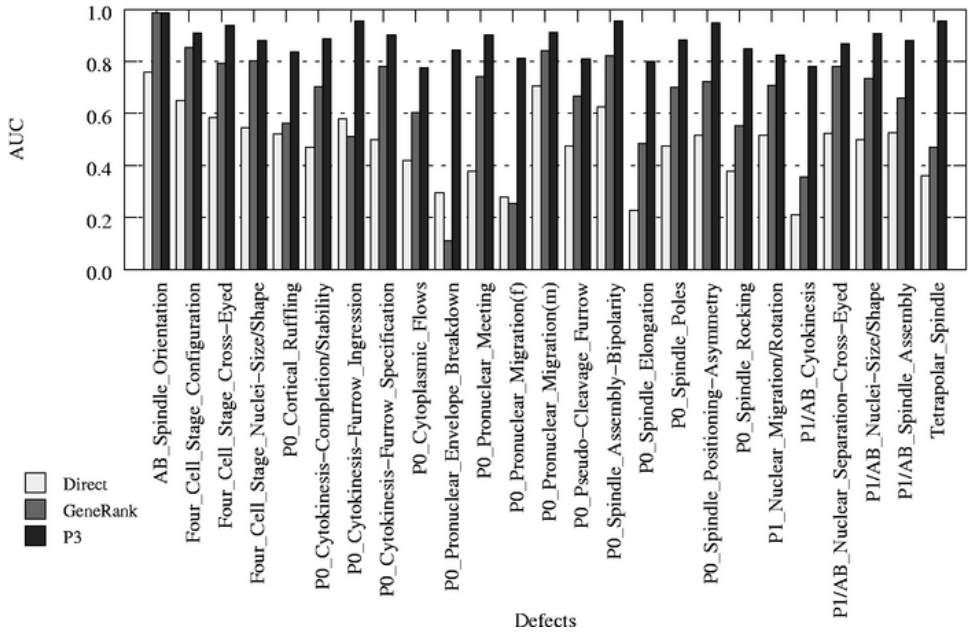


Figure 2.4: Overall performance comparison on the *C. elegans* dataset. Direct: ranking genes using the interaction network only; GeneRank: $d = 0.1$; P^3: $\beta = 0.6$

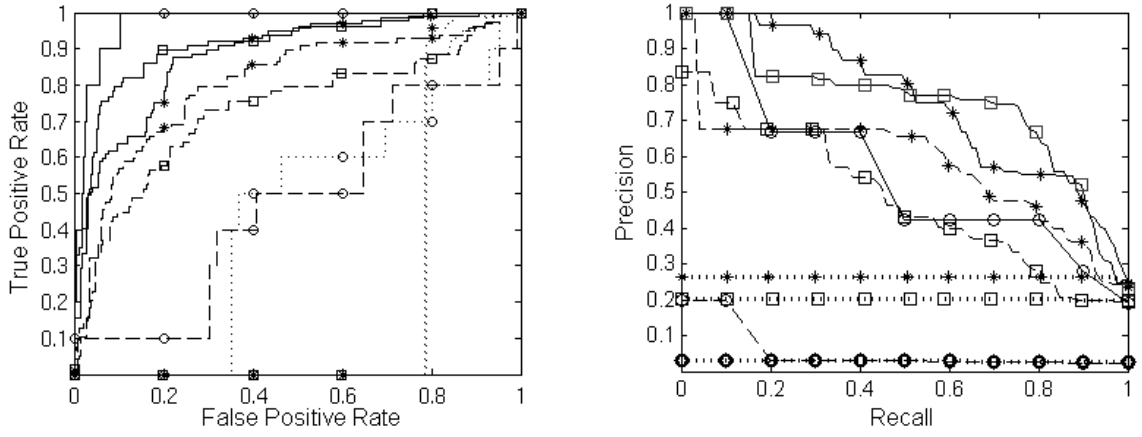


Figure 2.5: (left) ROC curves on *C. elegans*. (right) Precision vs. Recall on *C. elegans*. Direct: points, GeneRank: dashed line, P^3: solid line; square: P1-AB Nuclear-Size-Shape, star: Four Cell Stage Nuclei-Size-Shape, circle: Tetrapolar Spindle.

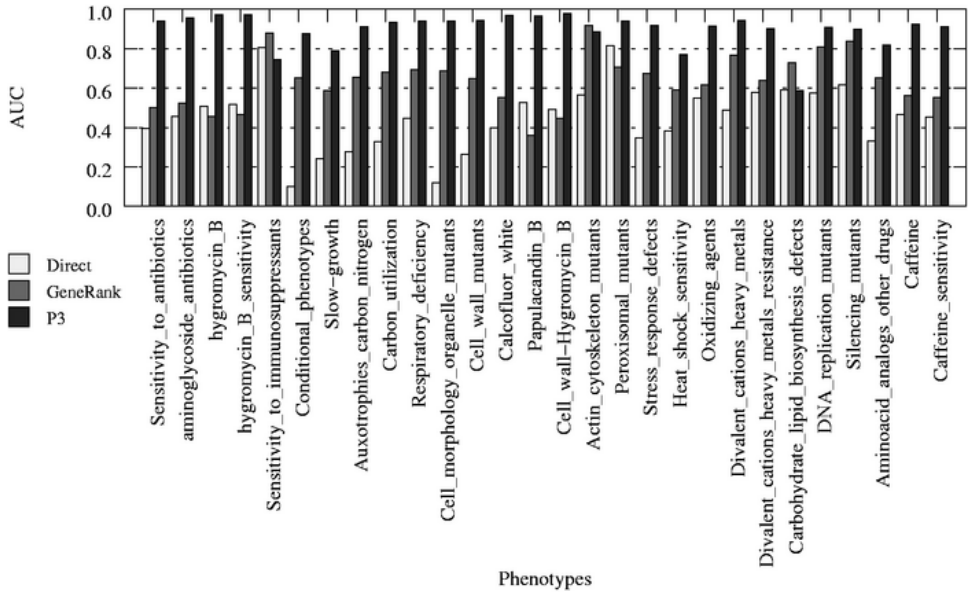


Figure 2.6: Overall performance comparison on yeast phenotype dataset. Direct: ranking genes using the interaction network only; GeneRank: $d = 0.1$; P^3: $\beta = 0.6$

2.4.3 Results

To evaluate the prediction performance for each phenotype, we used the leave-one-out and k-fold cross validation approaches [27, 81]. For the leave-one-out approach, one gene/phenotype pair is ignored from the original dataset each time, and the prediction algorithm is applied on the remaining dataset to see if that gene/phenotype pair is predicted correctly. Results show that our method outperforms the direct method and GeneRank method in almost all cases. We compared the Area Under the Receiver Operating Characteristic (AUC ROC) curve for each phenotype and plot the ROC curve and Precision-Recall curves for some phenotypes for further performance comparison. For k-fold cross validation, the original gene/phenotype pairs are separated into k groups, 10 in *C. elegans* and 5 in yeast, one of them is selected as test data and the remaining are used as training data. The distributions of AUC were compared. P^3 outperforms other methods in all cases. In the experiment of predicting phenotypes to genes without any interactions, results show that P^3 is able to predict a majority of these phenotypes with high accuracy.

Leave-one-out

C. elegans: For each phenotype prediction, we computed true-positive rate versus false-negative rate to measure the recovery of genes with the given phenotype. The comparison of the area under the Receiver Operating Characteristic curve for each phenotype is shown in Figure 2.4. For visualization purpose, 20 defects are randomly selected for discussion here. The defect "AB Spindle Orientation" shows the highest AUC in the results of all three methods, with values of 0.99

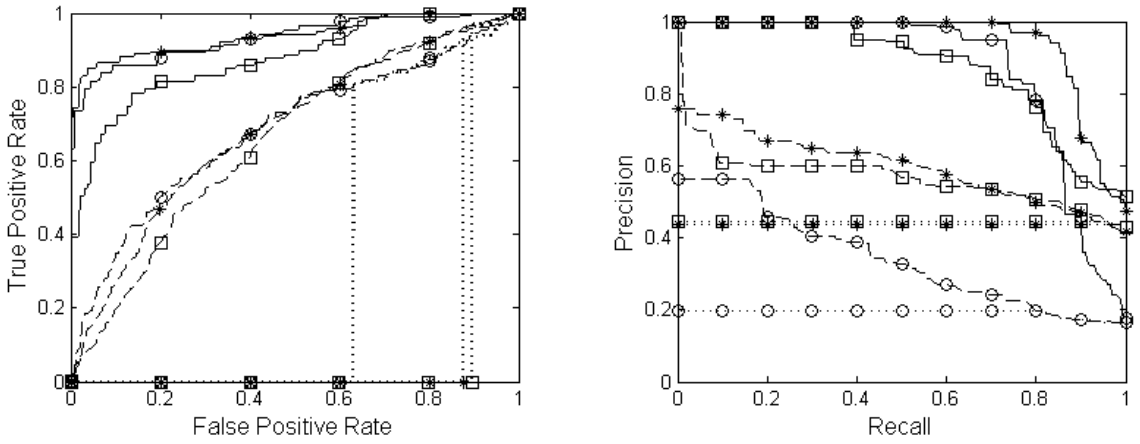


Figure 2.7: (left) ROC curves on yeast. (right) Precision vs. Recall on yeast. Direct: points, GeneRank: dashed line, P^3: solid line; circle: Carbon utilization, square: Conditional phenotypes, star: Cell morphology and organelle mutants.

in P^3 and GeneRank, and 0.76 in the Direct method. P^3 is always better than the Direct method and outperforms the GeneRank method in most cases. The AUCs of P^3 are greater than those of Direct method and GeneRank by 0.37 and 0.2, in average respectively, and the maximum differences are 0.6 and 0.73, respectively. Only three defects, "Egg Size/Shape", "AB Spindle Orientation" and "P1/AB Cortical Activity" show that GeneRank method is slightly better than P^3, with the maximum difference of AUC as 0.028. Three phenotypes, "Tetrapolar Spindle", "Four Cell Stage Nuclei-Size-Shape", and "P1-AB Nuclear-Size-Shape", that have both high AUC and precision-recall for P^3 were chosen for further comparison. Figure 2.5 (left) shows their ROC curves, and the corresponding precision-recall curves are shown in Figure 2.5 (right).

Yeast: Similar to the study in *C. elegans*, we computed true-positive rate versus false-negative rate and precisions at certain recall levels. The comparison of the area under the Receiver Operating Characteristic curve for each phenotype is shown in Figure 2.6. For simplicity, we show the results for 28 phenotypes among the 72 examined phenotypes. The highest AUC in the selected results of P^3 is 0.98, from "Cell wall-Hygromycin B", that of the Direct method is about 0.81, from "Peroxisomal mutants", and GeneRank has the highest AUC value about 0.88, from "Sensitivity to immunosuppressants". P^3 outperforms GeneRank and the Direct method in most cases. The AUCs of P^3 are greater than those of Direct method and GeneRank by 0.4 and 0.2 in average respectively, and the maximum differences are 0.6 and 0.8 respectively. Three phenotypes that have both high AUC values and precisions among the result of P^3 method were chosen for further comparison. They are "Conditional phenotypes", "Carbon utilization", and "Cell morphology and organelle mutants". Figure 2.7 (left) shows their ROC curves, and the corresponding precision-recall curves are shown in Figure 2.7 (right).

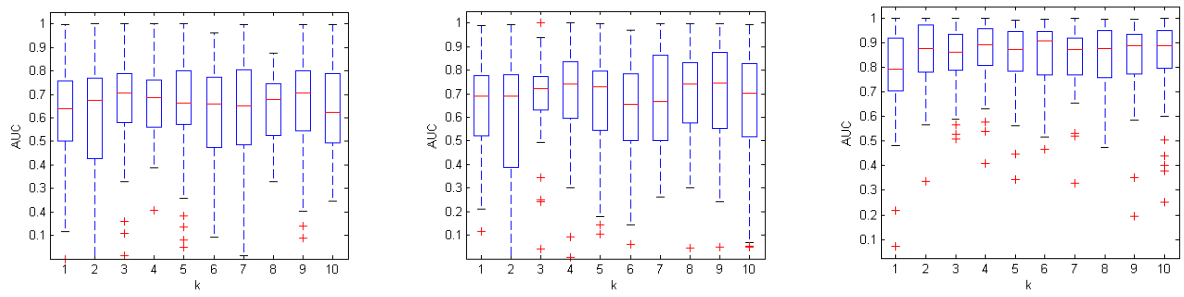


Figure 2.8: AUC distributions on *C. elegans*. Direct method (left), GeneRank method (middle), and P^3 (right).

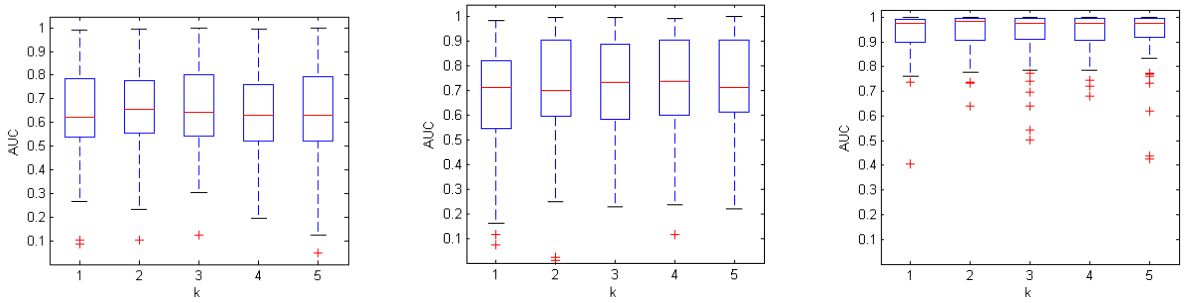


Figure 2.9: AUC distributions on yeast. Direct method (left), GeneRank method (middle), and P^3 (right).

k-fold cross validation

C. elegans. 10-fold cross validation was carried out on *C. elegans* data. Figure 2.8 shows the distributions of AUC values of each method. The median, lower quantile, and upper quantile of each group is plotted. As is evident, the performance is considerably improved by using P^3 for phenotype prediction.

Yeast. 5-fold cross validation was carried out on the yeast data. Figure 2.9 shows the comparison of distributions of AUC. The median, lower quantile, and upper quantile of each group is plotted. P^3 outperforms the other two methods in all cases.

Predicting Phenotypes to genes without any interactions

To evaluate our approach in predicting phenotypes for genes without any interaction information, we identified those genes that have at least two phenotypes but without interactions in both datasets. We used the phenotype graphs obtained in the leave-one-out experiment, that were derived without any information about the test genes. The target gene/phenotype pairs are separated almost equally

into two groups: one for training and another for testing. For example, for each gene, if it has two phenotypes then one is in the training group and another is in the test group. Results show that P^3 can predict most of the phenotypes successfully. Table 2.5 presents the characteristics of the data and results.

Table 2.5: Predicting phenotypes for genes without interactions.

Organism	Genes	Predicted with AUC ≥ 0.8
Caenorhabditis elegans	42	24
Saccharomyces cerevisiae	48	46

2.4.4 Phenotype Relations

The complete directed graph of phenotypes (representing the closeness matrix of the phenotypes) is too complex to describe in detail here. Therefore, we partition the graph into several highly connected subgraphs by using the CAST [5] algorithm. CAST is a heuristic approach for solving the 'corrupted cliques' problem. It transforms an undirected graph into a set of cliques or almost cliques by repetitively adding nodes having maximum average similarity to the current clique, as long as the similarity is above a threshold λ, and removing nodes with minimum average similarity to the clique, when the similarity is less than the threshold. The process stops when there are no more nodes to add or remove. First, directions are removed from the edges in the original phenotype graph. For each pair of phenotypes, two directed edges are merged into one undirected edge. Every new edge is assigned a new weight that is the average of weights of the original two edges associated to the two phenotypes. The graph is further adjusted by deletions of "weak" connections between phenotypes. For example, if the weight of the connection between phenotype p to q is less than a threshold t, then the corresponding edge is removed. We run the CAST algorithm on this simplified graph. A set of cliques and almost cliques is obtained. Each clique/almost clique is a cluster of single or a set of highly related phenotypes. Genes causing these phenotypes tend to interact or function together. Figure 2.10 and Figure 2.11 show some of the phenotype cliques obtained. The thickness of links represents the closeness between phenotypes. Multiple values are used for parameters t and λ. As t and λ decrease, the number of cliques/almost cliques decreases and the size of the maximum clique/almost clique increases. We choose the parameter values that give small cliques so that they are relatively easy to interpret biologically. In *C. elegans*, there are 23 cliques/almost cliques, and the largest clique/almost clique contains 11 nodes, one clique with 5 nodes, 4 nodes, and 3 nodes respectively, three cliques with 2 nodes, and the rest are singletons. In yeast, there are 41 cliques/almost cliques, and the largest clique contains 11 nodes, one clique with 4 nodes, six with 3 nodes, and six with 2 nodes; the remaining are singletons.

The *C. elegans* phenotypes identified in Figure 2.10 are all related to cell division. The edges suggest that there are distinct relationships between the formation and behavior of the nuclei,

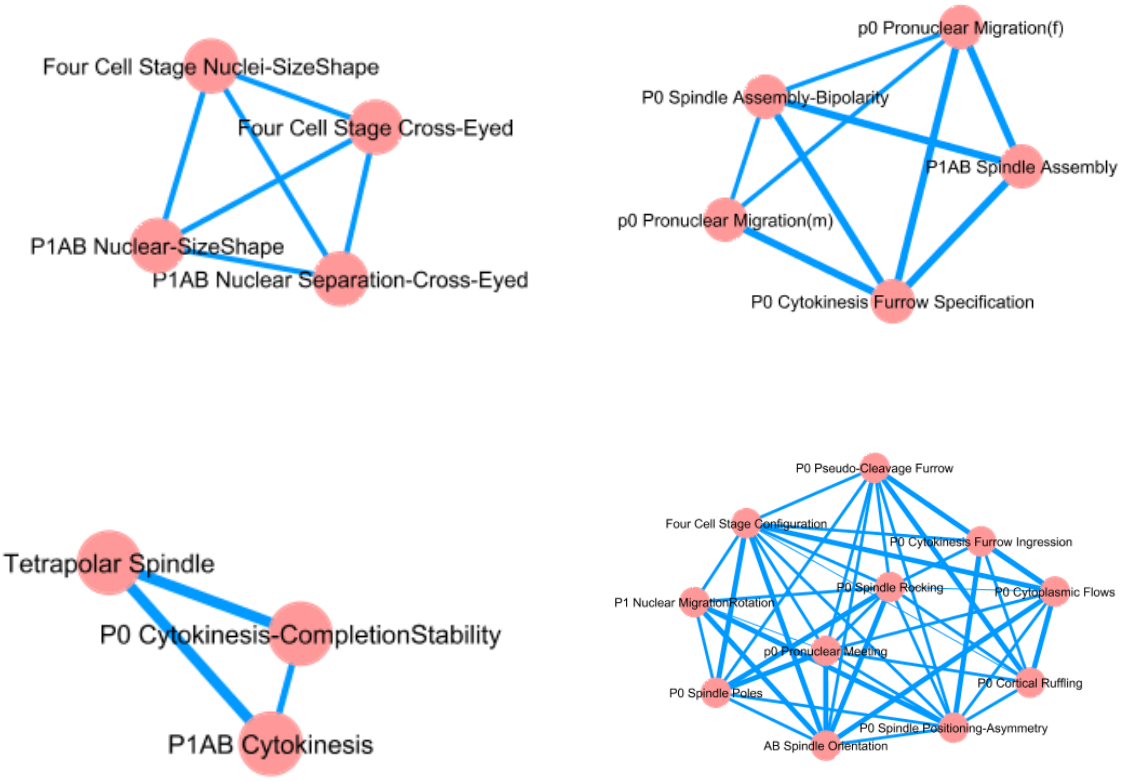


Figure 2.10: Phenotype cliques in the *C. elegans* dataset derived from P^3.

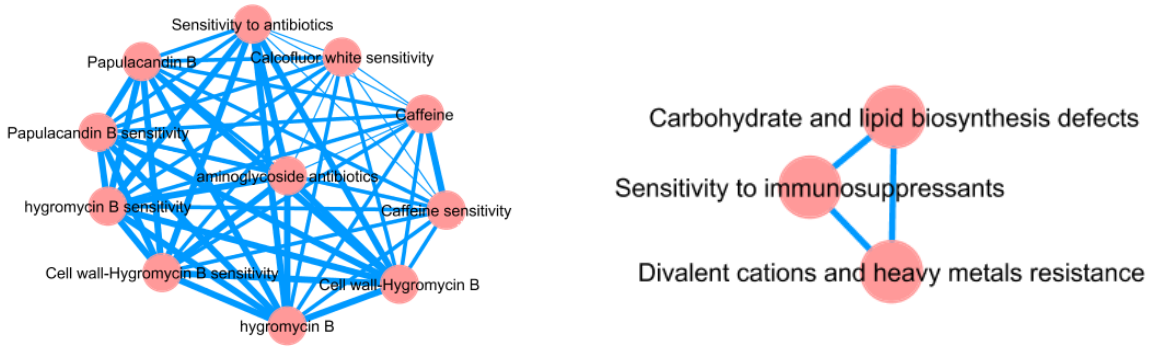


Figure 2.11: Phenotype cliques in the *S. cerevisiae* dataset derived from P^3.

indicative of a functional role for structural proteins. The role of structural proteins, acting as conduits for macromolecular and organellar movement, also can be seen in the largest clique where cytokinesis (splitting of the cytoplasm to form two cells) and furrow formation (where the cells are divided in half) are related.

The larger yeast cliques in Figure 2.11 pertains to drug sensitivities, including antibiotics. Such associations could potentially be reflective of the role of the extracellular domain in resistance or non-resistance to select antibiotics. Caffeine sensitivity has been related to the synthesis of phospholipids (cell membrane components) and changes in calcium flux. Indeed, the smaller clique relates all of these concepts through sensitivity to immunosuppresants, a sensitivity that is related to phosphorylation-based signal transduction cascades.

2.5 Discussion

We have presented an approach to modeling phenotype relations and using these relationships to predict phenotypes for uncharacterized genes. The strong results indicate that the combination of gene networks and phenotype profiles provides a powerful synergy that is not obtainable with either method alone. One limitation is that to be able to make predictions, a gene should have at least one known phenotype.

In understanding the sucess of our framework, there are three key questions that arise:

1. Can we formally prove the convergence of iterations using Eqn. 2.2?
2. How crucial is the notion of relative importance (vis-a-vis global importance) in inferring relationships between phenotypes?
3. If relative importance is critical, how sensitive is the algorithm to the setting of the initial ranks in Eqn. 2.1?

We proceed to answer these questions below.

2.5.1 Proof of Convergence

First, we study the convergence properties of our algorithm. We prove that it converges under conditions based on the Perron-Frobenius theorems and the theory of Markov chains. The convergence property of our algorithm also can be inferred from previous studies on the convergence of PageRank and similarly motivated algorithms [10]. The related definitions are given below, which are drawn from [6, 26, 30, 40].

Definition 2.5.1. *A matrix M is non-negative if every entry $m_{ij} \geq 0$. Similarly, a vector is non-negative if every entry is non-negative.*

Definition 2.5.2. *A square matrix is right/left stochastic if every entry has a non-negative value and the sum of each row/column of the matrix is 1.*

Definition 2.5.3. *A square matrix A is non-singular if there is another matrix B such that $AB = I$, where I is the identity matrix. B is also written as A^{-1}.*

Definition 2.5.4. *Let A be a matrix. If there is a non-zero vector $\vec{x} \in R^n$ such that $A\vec{x} = \lambda\vec{x}$ for some scalar λ, then λ is called an eigenvalue of A with corresponding (right) eigenvector $\vec{x}$. The eigenvalue with the largest magnitude is called the dominant eigenvalue of A.*

Definition 2.5.5. *A non-negative square matrix M is irreducible if and only if for any pair of indices (i and j) there is an positive integer t such that $(M^t)_{ij} > 0$.*

If an undirected graph is connected, i.e., for every pair of vertices there exists a path between them, then the adjacency matrix of this graph is irreducible.

We now turn to some definitions involving Markov chains [54]. Every gene in the network can be considered as a *state*, i.e., the gene set G is the *state space*. First, we define $\mathcal{X} = \{X_i \mid 0 \leq i\}$ to be a set of random variables over same probability space, where $i = 0, 1, 2, ...$ (non-negative integers) denotes the discrete time step, and X_i indicates the state of the system at time i. If the future state of the system is only influenced by its current state, then the system is said to have the *Markov property*. More precisely,

$$\mathbf{P}r(X_{i+1} \mid X_0, X_1, ..., X_i) = \mathbf{P}r(X_{i+1} \mid X_i).$$

A Markov chain has *stationary transition prabability* or is said to be *homogeneous on time*, if for all $l, k \geq 0$ and for all $g_i, g_j \in G$, it satisfies

$$\mathbf{P}r(X_{l+1} = g_i \mid X_l = g_j) = \mathbf{P}r(X_{k+1} = g_i \mid X_k = g_j).$$

The conditional probability

$$t_{j,i} = \mathbf{P}r(X_{l+1} = g_i \mid X_l = g_j)$$

is called the transition probability of this Markov chain. Let the size of the state space is s, then the transition matrix is a $s \times s$ matrix as follows:

$$T = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,s} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,s} \\ \cdots & \cdots & \cdots & \cdots \\ p_{s,1} & p_{s,2} & \cdots & p_{s,s} \end{pmatrix}$$

T is a right stochastic matrix based on Definition 2.5.2.

Definition 2.5.6. *The period of a state X_i is k if any return to state X_i must occur in multiples of k time steps. A Markov chain is called aperiodic if all of its states have period 1.*

Notice that a Markov chain is aperiodic if and only if there is no integer $k > 1$ that divides the length of every cycle of the directed graph associated with the transition matrix. The gene network in this study can be interpreted as a directed graph by replacing each edge with two directed edges pointing to two end nodes respectively. In the following, we call a network *aperiodic* if the corresponding directed graph has the above property.

Definition 2.5.7. *A Markov chain is irreducible if its transition matrix is irreducible.*

Definition 2.5.8. *The vector π is called a stationary distribution of the chain, if π has entries $\pi_i, i \in \mathcal{S}$, ($\mathcal{S}$ is the state space), such that*

1. $\pi_j \geq 0$ *for all* j, *and* $\sum_j \pi_j = 1$.
2. $\pi = \pi T$, *i.e.,* $\pi_j = \sum_i \pi_i T_{ij}$ *for all* j.

Theorem 2.5.9. *([30] p.66) Let $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ be the eigenvalues of a square matrix A. If the dominant eigenvalue of A, i.e., $\max_{1 \leq i \leq n} |\lambda_i|$, is less than 1, then*

1. $I - A$ *is non-singular, and*
2. $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$, *($A^0 = I$).*

Theorem 2.5.10. *([6] p.49) The dominant eigenvalue of a stochastic matrix is 1.*

Theorem 2.5.11. *([54] see page.73) If a finite Markov chain is irreducible and aperiodic, then $\lim_{k \to \infty} \pi_0 T^k = \pi$, where T is the transition matrix, π_0 is the initial probability distribution. The Markov chain has a unique stationary distribution, π.*

Let $W_j = \sum_{i=1}^{n} w_{ji}$ be the total weight of interactions involving gene g_j. Without loss of generality, we assume that the gene network does not contain singletons, i.e., a single node without any adjacent edges. We define an $n \times n$ left stochastic matrix $\mathcal{T}$ as below:

$$\mathcal{T}_{ij} = \begin{cases} \frac{w_{ji}}{W_j} & \text{if } W_j > 0\,, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

where the sum of each column equals to 1. $\mathcal{T}$ is the normalized adjacency matrix of the network, i.e., dividing each cell with the sum of the corresponding column. Note that this matrix is completely dependent on the weights of interactions, which do not change in the whole process. The equation for iteratively computing the ranks shown in Equation 2.2 can be written in the vector format as follows:

$$\vec{r_p}^{\,k} = \beta \vec{r_p}^{\,0} + (1 - \beta)\mathcal{T}\vec{r_p}^{\,k-1}. \tag{2.7}$$

where p denotes the phenotype.

Theorem 2.5.12. *When $0 < \beta \leq 1$, the proposed algorithm presented by Equation 2.7 converges. When $\beta = 0$, the algorithm converges, if all connected components of the associated network are aperiodic.*

Proof. 1. When $\beta = 1$, there is no iterative part in the equation, i.e., the initial ranks will be the solution, and so the algorithm converges.

2. Consider $0 < \beta < 1$. In order to prove that Equation 2.7 converges, we have to show that $\vec{r_p}$ exists such that

$$\vec{r_p} = \beta \vec{r_p}^0 + (1 - \beta)\mathcal{T}\vec{r_p}. \tag{2.8}$$

$\Rightarrow$

$$(I - (1 - \beta)\mathcal{T})\vec{r_p} = \beta \vec{r_p}^0. \tag{2.9}$$

where I is the identity matrix. Let $C = (1 - \beta)\mathcal{T}$. The dominant eigenvalue of $\mathcal{T}$ is 1 based on Theorem 2.5.10, so the dominant eigenvalue of C is $1 - \beta < 1$. According to Theorem 2.5.9, $I - C$ is non-singular and

$$(I - C)^{-1} = I + C + C^2 + C^3 + ...$$

Thus Equation 2.9 reduces to

$$\vec{r_p} = \beta \vec{r_p}^0 (I - C)^{-1}.$$

Assuming λ is an eigenvalue of C and x is the corresponding eigenvector, then

$$(I + C + C^2 + C^3 + ...)x = (1 + \lambda + \lambda^2 + \lambda^3 + ...)x$$

The absolute value of λ should be less than or equal to the dominant eigenvalue of C, so $|\lambda| < 1$. Therefore, $1 + \lambda + \lambda^2 + \lambda^3 + ... + \lambda^i + ...$ converges when $i \to \infty$. Therefore, the algorithm converges to $\vec{r_p}$.

3. When $\beta = 0$, Equation 2.7 reduces to

$$\vec{r_p}^k = \mathcal{T}\vec{r_p}^{k-1}. \tag{2.10}$$

$\Rightarrow$

$$(\vec{r_p}^k)^T = (\vec{r_p}^{k-1})^T \mathcal{T}^T. \tag{2.11}$$

$\mathcal{T}^T$ is a right stochastic matrix. The above equation represents a finite Markov chain with transition matrix $\mathcal{T}^T$.

(a) If the network is connected, then the transition matrix $\mathcal{T}^T$ is irreducible based on Definitions 2.5.5 and 2.5.7. The associated Markov chain is also irreducible. According to Theorem 2.5.11, the algorithm converges when the Markov chain is aperiodic. It also means that the underlying network (presented as a directed graph) should be aperiodic. If the network is not aperiodic, then the Markov chain may not converge. For example, a periodic network could contain only two nodes and one edge, with the transition matrix given by

$$T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

In this case, as $n \to \infty$, the powers T^i approach two different matrices:

$$T^n = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, n = 2m, \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, n = 2m+1, m \text{ is a natural number.} \end{cases}$$

therefore, this Markov chain does not converge.

(b) If the network contains multiple connected components, the indices of the transition matrix can be arranged such that the transition matrix is in the form of a block diagonal matrix as below:

$$T = \begin{pmatrix} T_{11} & 0 & 0 & 0 \\ 0 & T_{22} & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & T_{kk} \end{pmatrix}$$

where k is the number of connected components. Each block $T_{jj}, 1 \leq j \leq k$ is the transition matrix of the corresponding connected component. Thus

$$T^n = \begin{pmatrix} T_{11}^n & 0 & 0 & 0 \\ 0 & T_{22}^n & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & T_{kk}^n \end{pmatrix}$$

Hence, as $n \to \infty$, T^n converges when each diagonal block converges. The above conclusion can be applied to each block matrix. $\lim_{n=1}^{\infty} T_{ii}^n$ exists when the underlying subnetwork (connected component) is aperiodic. Therefore, when $\beta = 0$, the algorithm converges, if all connected components of the associated network are aperiodic.

□

2.5.2 Relative Levels of Relative Importance

In this section, we study the improvement afforded by the relative importance method as compared to a global importance formulation. Specifically, we study the effect of two different ways of setting the prior probabilities (which controls the balance between relative-vs-global) and study their effects on prediction performance. This strategy is implemented by designing an interpolation scheme that smoothly morphs from the original relative importance method on the one hand, and the global importance method on the other.

There are two ways this interpolation can be realized. In the first case, we use a parameter α to set up the prior probabilities as:

$$r^0_{g_i} = \begin{cases} \frac{1}{\alpha n + (1-\alpha)\|R\|,} & \text{if } g_i \in R \\ \frac{\alpha}{\alpha n + (1-\alpha)\|R\|} & \text{otherwise.} \end{cases} \tag{2.12}$$

where n is the number of total genes in the network and $\parallel R \parallel$ denotes the size of the root set R. Notice that the sum of the total prior probabilities is 1. When $\alpha = 0$, observe that Equation 2.12 reduces to Equation 2.1, our original formulation. When $\alpha = 1$, every gene (including genes in the root set) is assigned with prior probability $1/n$.

In the second interpolation scheme, α is used as follows:

$$r^0_{g_i} = \begin{cases} \frac{1-\alpha}{\|R\|} + \frac{\alpha}{n} & \text{if } g_i \in R \\ \frac{\alpha}{n} & \text{otherwise.} \end{cases} \tag{2.13}$$

When $\alpha = 0$, Equation 2.13 reduces to Equation 2.1 as before. When $\alpha = 1$, every gene is assigned an equal prior probability $1/n$.

Thus, both schemes interpolate between the same extremes but different in how they raise the influence of relative importance.

Table 2.6: Results of first interpolation scheme between relative and global importance methods on the *C. elegans* dataset ($\beta = 0.5$).

	AB Sppindle Orientation	*Tetrapolar Spindle*	*P0 Cytokinesis Furrow Ingression*	*Meiotic Arrest*	*P0 Pronuclei Number*	*P1-AB Nuclear-Number*
#Gene	4	10	8	54	79	82
1.0	0.1496	0.1722	0.1763	0.7056	0.7523	0.6712
0.9	0.1797	0.1951	0.1851	0.7726	0.7688	0.7062
0.8	0.2025	0.2334	0.2039	0.8323	0.7826	0.7263
0.7	0.2933	0.2920	0.2239	0.8791	0.7899	0.7447
0.6	0.3630	0.3710	0.2464	0.9130	0.7981	0.7598
0.5	0.5210	0.5002	0.2894	0.9172	0.8036	0.7667
0.4	0.7356	0.6468	0.3674	0.9211	0.8111	0.7741
0.3	0.9261	0.8051	0.5549	0.9237	0.8127	0.7740
0.2	0.9874	0.9105	0.7703	0.9222	0.8118	0.7750
0.1	0.9886	0.9590	0.8953	0.9245	0.8076	0.7733
0.0	0.9880	0.9710	0.9178	0.9276	0.8070	0.7726

Table 2.7: Results of second interpolation scheme between relative and global importance methods on the *C. elegans* dataset ($\beta = 0.5$).

	AB Sppindle Orientation	*Tetrapolar Spindle*	*P0 Cytokinesis Furrow Ingression*	*Meiotic Arrest*	*P0 Pronuclei Number*	*P1-AB Nuclear-Number*
#Gene	4	10	8	54	79	82
1.0	0.1496	0.1722	0.1763	0.7056	0.7523	0.6712
0.9	0.9880	0.9354	0.8796	0.9167	0.7979	0.7582
0.8	0.9880	0.9637	0.9087	0.9227	0.8082	0.7719
0.7	0.9880	0.9678	0.9147	0.9227	0.8127	0.7745
0.6	0.9880	0.9688	0.9166	0.9229	0.8123	0.7748
0.5	0.9880	0.9695	0.9160	0.9231	0.8123	0.7733
0.4	0.9880	0.9702	0.9163	0.9261	0.8099	0.7734
0.3	0.9880	0.9702	0.9163	0.9269	0.8068	0.7735
0.2	0.9880	0.9707	0.9163	0.9269	0.8072	0.7732
0.1	0.9880	0.9710	0.9169	0.9273	0.8068	0.7730
0.0	0.9880	0.9710	0.9178	0.9276	0.8070	0.7726

Table 2.8: Results of first interpolation scheme between relative and global importance methods on the *S. cerevisiae* dataset ($\beta = 0.5$).

	Respiratory deficiency	*Hygromycin B*	*Actin cytoskeleton mutants*	*Silencing mutants*	*DNA replication mutants*
#Gene	153	25	48	26	31
1.0	0.0870	0.1287	0.6459	0.8278	0.8301
0.9	0.1184	0.1447	0.6641	0.8357	0.8356
0.8	0.1919	0.1730	0.6882	0.8469	0.8406
0.7	0.3858	0.2206	0.7197	0.8653	0.8444
0.6	0.7154	0.3341	0.7519	0.8794	0.8534
0.5	0.8975	0.5858	0.8004	0.8945	0.8608
0.4	0.9575	0.8719	0.8602	0.9119	0.8703
0.3	0.9801	0.9856	0.9051	0.9299	0.8843
0.2	0.9866	0.9911	0.9275	0.9455	0.8965
0.1	0.9877	0.9912	0.9413	0.9650	0.9132
0.0	0.9879	0.9926	0.9501	0.9770	0.9245

Table 2.9: Results of second interpolation scheme between relative and global importance methods on the *S. cerevisiae* dataset ($\beta = 0.5$).

	Respiratory deficiency	*Hygromycin B*	*Actin cytoskeleton mutants*	*Silencing mutants*	*DNA replication mutants*
#Gene	153	25	48	26	31
1.0	0.0870	0.1287	0.6459	0.8278	0.8301
0.9	0.8714	0.9923	0.9164	0.9568	0.9028
0.8	0.9754	0.9923	0.9373	0.9686	0.9145
0.7	0.9857	0.9923	0.9434	0.9713	0.9181
0.6	0.9874	0.9923	0.9451	0.9732	0.9211
0.5	0.9876	0.9924	0.9473	0.9740	0.9227
0.4	0.9878	0.9925	0.9490	0.9754	0.9231
0.3	0.9878	0.9926	0.9497	0.9762	0.9238
0.2	0.9878	0.9926	0.9498	0.9766	0.9240
0.1	0.9878	0.9926	0.9499	0.9769	0.9241
0.0	0.9879	0.9926	0.9501	0.9770	0.9245

We ran the algorithm with the two modified prior probability distributions on both the *C. elegans* and *S. cerevisiae* datasets. In both cases, α was varied from 0 to 1 with step size 0.1. We compared the area under the ROC curve for different phenotypes with different settings of α. Table 2.6 and Table 2.8 demonstrate the results of the first interpolation scheme on the *C. elegans* and *S.*

cerevisiae datasets, respectively. Table 2.7 and Table 2.9 demonstrate the results of the second interpolation scheme on the *C. elegans* and *S. cerevisiae* datasets, respectively. The results show that any amount of relative importance weighting renders the algorithm more effective than the global importance method, for both interpolation schemes. Furthermore, the performance increases with the increase in the amount of bias afforded to the root set, with the first interpolation scheme demonstrating a more aggressive performance improvement than the second.

Now that we have established the role of the relative importance method, we turn to significance evaluation of the actual root sets involved. For each phenotype, we randomly select a set of genes from the entire gene network as the root set for computing the ranks of genes for that phenotype. The size of this randomly selected root set is maintained the same as the number of genes known to give rise to that phenotype. Then we compute the *p*-value of the rank performance (i.e., AUC value) with random root set greater than that of the original root set. All phenotypes with original root sets have *p*-value much less than 10^{-3}. This demonstrates that the performance of our algorithm is statistically significant and that the ranking of genes for a phenotype is highly dependent on the local region of genes known to associate with that phenotype.

2.5.3 Optimizing Initial Ranks

The third question that emerges from the above discussion is the importance of setting the 'right' values for the prior probabilities. In our discussion so far, every gene in the root set is given the same, uniform, prior probability. However, it may be argued that a particular subset of the root set may play a more important role for an accurate prediction than the remainder of the root set. An interesting question is if further tuning of these prior probabilities can lead to tangible improvements in performance.

We define an optimization problem to address this question. Let $\vec{r_p}^0$ be the vector of prior probabilities (initial ranking) for a given phenotype p. Recall that the probabilities of all genes are computed iteratively as follows:

$$\vec{r_p}^k = \beta \vec{r_p}^0 + (1-\beta)\mathcal{T}\vec{r_p}^{k-1}. \tag{2.14}$$

where k is the number of iterations, and $\mathcal{T}$ is the transition matrix. Equation 2.14 can be re-written by extending the recursion implicit in $\vec{r_p}^{k-1}$ down to $\vec{r_p}^0$:

$$\begin{aligned} \vec{r_p}^k &= \beta\vec{r_p}^0 + \beta(1-\beta)\mathcal{T}\vec{r_p}^0 + \beta(1-\beta)^2\mathcal{T}^2\vec{r_p}^0 + \cdots + \beta(1-\beta)^{k-1}\mathcal{T}^{k-1}\vec{r_p}^0 + (1-\beta)^k\mathcal{T}^k\vec{r_p}^0 \\ &= (\beta I + \beta(1-\beta)\mathcal{T} + \beta(1-\beta)^2\mathcal{T}^2 + \cdots + \beta(1-\beta)^{k-1}\mathcal{T}^{k-1} + (1-\beta)^k\mathcal{T}^k)\vec{r_p}^0 \\ &= \mathcal{A}\vec{r_p}^0. \end{aligned} \tag{2.15}$$

where I is the identity matrix and $\mathcal{A}$ denotes the right part of the previous equation except $\vec{r_p}^0$. Previously, we have proved that the algorithm will converge, so we can bound the number of

iterations k by a sufficiently large integer, and thus treat $\mathcal{A}$ as a constant matrix. Given a phenotype p, we need to compute the prior probabilities $\vec{r_p}^0$ that will produce the best ranking prediction. When used with our training data, note that $\vec{r_p}^k$ is provided only as ranks, not as absolute values. Hence this problem is more complicated than that of solving a traditional linear system.

As an aside, this type of problem can be interpreted as an 'inverse information retrieval' problem where $\mathcal{A}$ can be viewed as a term-document matrix representing a collection, $\vec{r_p}^0$ represents the input query (in term space), and $\vec{r_p}^k$ denotes the final ranking (of documents). We seek to identify the right query that will produce a desired ranking of documents.

In our gene perturbation context, the motivations are similar. For every pair of genes g_i and g_j in the root set R_p of phenotype p, if it is known from the given data that $r_{g_i} < r_{g_j}$, then we want the predicted rankings to follow the same order, i.e., $\hat{r_{g_i}} < \hat{r_{g_j}}$. (An example of how such rankings can be obtained is by ordering genes according to the reproducibilities of observing the phenotype after knocking them down.)

We now show how computing the desired prior probabilities can be formulated as a semi-definite programming (SDP) problem. First, we present some background about SDP. The problem of optimizing a linear function over the intersection of the cone of positive semidefinite matrices with an affine space is called *semidefinite programming* (SDP) [84]. A standard formula is described below:

Given:

$$A_k,\ C \in R^{n \times n},\ A_k \text{ is a symmetric matrix } \ b_k \in R,\ k = 1, 2, ..., m$$

Objective function and constraints:

$$\begin{array}{ll} \text{minimize} & C \bullet X \\ \text{such that} & A_k \bullet X = b_k, k = 1, ..., m \\ & X \geq \mathbf{0} \end{array}$$

where $X \in R^{n \times n}$ is the variable. The operation $P \bullet Q$ represents the trace inner product of two matrices P and Q.

A second notion we will find useful is *mixed semidefinite-quadratic-linear programs*. These are linear optimization problems over a product of semidefinite cones, quadratic cones, and the non-negative orthant. Together, these cones make up all possible homogeneous self-dual cones over the reals. SDP-Pack [1] is a software package to solve such programs. The standard formula of a mixed semidefinite-quadratic-linear program is as follows:

Given data:

$$(A_S)_k, C_S \in R^{n_S \times n_S},\ \ C_S \text{ and } (A_S)_k \text{ are diagonal matrices;}$$

$$\vec{c_Q}, (\vec{a_Q})_k \in R^{n_Q}, \vec{c_L}, (\vec{a_L})_k \in R^{n_L} \text{ are vectors, } b_k \in R,\ k = 1, 2, ..., m$$

Objective function and constraints;

$$\begin{aligned}
\text{minimize} \quad & C_S \bullet X_S + \vec{c_Q}^T \vec{x_Q} + \vec{c_L}^T \vec{x_L} \\
\text{such that} \quad & (A_S)_k \bullet X_S + (\vec{a_Q})_k^T \vec{x_Q} + (\vec{a_L})_k^T \vec{x_L} = b_k,\ k = 1, ..., m \\
& X_S \succeq \mathbf{0} \\
& (x_Q)_1 \geq \sqrt{\sum_{j=2}^{n_Q} (x_Q)_j^2} \\
& (x_L)_j \geq 0, j = 1, ..., n_L.
\end{aligned}$$

where X_S is the positive semidefinite symmetric matrix varaible, $\vec{x_Q}$ and $\vec{x_L}$ are vector variables.

As described earlier in Equation 2.15, for a given phenotype p, $\vec{r_p}^0$ is the optimal initial ranks we are trying to find. Let $\vec{p}$ be a binary vector, where a cell with value 1 means that the corresponding gene is known to give rise to p, otherwise 0. The problem of learning the optimal initial ranks can be formulated as a mixed semidefinite-quadratic-linear programming problem with following mappings:

- Let $\vec{r_p}^0 = X_S \vec{p}$ where $X_S \in R^{n \times n}$ is a symmetric matrix variable.
- Let $\vec{c_L}$ and $\vec{x_L}$ be one element vectors, with values $\vec{c_L} = [1]$, $\vec{x_L} = [t]$.
- Let $\vec{x_Q}$ be a vector variable with elements $[t, (\mathcal{A} X_S \vec{p} - \vec{y})]$, where $\vec{y}$ is a vector variable with a length of n. $\vec{y}$ is an auxiliary variable used to enforce that the predicted ranking is consistent with given rankings. In other words, the elements of $\vec{x_Q}$ are t and followed by the elements in the vector $(\mathcal{A} X_S \vec{p} - \vec{y})$.
- Let $k = 1$, A_S, C_S be zero matrices, $\vec{a_Q}$, $\vec{c_Q}$, and $\vec{a_L}$ be zero vectors.

The optimization formula can then be stated as:

$$\begin{aligned}
\text{minimize} \quad & t \\
\text{such that} \quad & \| \mathcal{A} X_S \vec{p} - \vec{y} \|_F \leq t, \\
& X_S \succeq \mathbf{0}, \\
& \sum_{i=1}^{n} (X_S \vec{p})_i = 1, \\
& \vec{y_i} \geq 0, \ \text{for all } 1 \leq i \leq n, \text{ and} \\
& \vec{y_i} \geq \vec{y_j} + \epsilon, \ \text{for all } 1 \leq i, j \leq n, i \neq j, \ \text{if } g_i \text{ ranked higher than } g_j \text{ in the given data.}
\end{aligned}$$

where ϵ is a non-negative small constant and $\| \dot{} \|_F$ denotes the Frobenius matrix norm. The first constraint maps to the quadratic constraint in the standard formula 2.16. The third constraint says that the sum of the prior probabilities should be 1.

We have solved this mixed semidefinite-quadratic-linear programming problem using SDP-Pack [1] and compared the performance improvement obtained by optimizing initial ranks versus using the original uniform assignment. Table 2.10 outlines experimental results on the *C.elegans* RNAi-induced early embryo defects dataset [80] and the *S.cerevisiae* knockout dataset [60]. Each dataset is separated into training and test sets. The training data is used to compute the optimal prior probabilities which then are used to predict the rankings of genes for phenotypes in the test set. Multiple ranking correlation metrics, such as Kendall's tau [50], Spearman's rho [41], and degree of agreement [77], were used to assess the results. The input of these correlation metrics are two ranking lists, one is the known gene-phenotype relation, another is computed using P^3 with the optimal initial rank. The results indicate that optimizing prior probabilities marginally improves the prediction accuracy beyond the vanilla uniform weighting scheme originally used.

These results mirror observations made by Ding et al. [18] who show that (web search) rankings obtained by using power iteration algorithms such as HITS, PageRank (and our algorithm as well) are actually quite similar to those obtained by ranking using indegree or outdegree metrics.

Table 2.10: Performance improvement with optimized prior probabilities.

Dataset	Kendall's tau	Spearman's rho	Degree of agreement
C.elegans	3%	9%	5%
S.cerevisiae	4%	6%	4%

Chapter 3

Constrained Graph Minimal Separator Mining

In this chapter, we introduce the problem of finding *constrained minimal separators* in interaction networks and show how closed itemset mining algorithms can be adapted to find desired separators. This work finds application in identifying sets of genes to knockdown for use in RNA interference or other gene perturbation studies. Experimental results on *S. cerevisiae* and *C. elegans* interaction networks demonstrate the applicability of this approach.

3.1 Introduction

Biology has entered the "network" era where, with the aid of high-throughput technologies, we are now able to analyze entire systems by taking into account complex and multi-faceted interactions between molecules. Investigating the structure and function of biological systems by gene perturbations using RNA interference technology is a case in point. RNA interference is a post-transcriptional gene perturbation approach, so that transcription proceeds unhindered but conversion of mRNA molecules to proteins (i.e., translation) is disrupted, potentially triggering a cascade of events, which in turn cause changes in phenotypes. In contrast to deletion mutants ("knockouts") using homologous recombination that interrupt gene function at the level of transcription itself, RNA interference is sometimes referred to as a "knockdown", since mRNA degradation is an imperfect process and some basal level of translation might be retained. Nevertheless RNA interference has proved to be a powerful, high-throughput, approach to elucidate gene function. Because the effect of a perturbation in levels of protein products is closely related to how a gene is situated in the larger network of interactions, the choice of which gene to knockdown must take into account not only the direct effects of such a knockdown but also the other subsystems indirectly influenced by the knockdown.

To model the relationship between gene perturbations and the resulting phenotypes, with an eye

toward capturing the complexity of gene interactions, we can formally study the effects of a perturbation using the terminology of graph separators [73]. In other words, to study the network effects of perturbation of a given gene or a set of genes, we investigate the sub-networks or sub-processes that the selected gene(s) separate. More interestingly, we can study the reverse problem of gene perturbation design: Which genes should be knocked down in order to separate targeted sub-networks from others with minimum side effects? In this chapter, we present an algorithm to address precisely this question by leveraging recent results [9] on the connection between graph separators and closed itemset mining.

3.2 Related Work

Finding minimal separators has important applications in biology and has been studied in different contexts. The minimal knockout problem, studied by Ruths *et al.* [74], seeks a minimum set of molecules to remove from a biological network such that some set of molecules are disconnected from the network but the other parts remains connected. In [74], it is shown that this problem is NP-hard using a reduction from the minimum set cover problem. This work also proposes a heuristic approach to find an approximate solution. It first constructs a candidate set that contains a knockout set. Second, a randomized search over all nodes in the candidate set is performed by iteratively removing a node until the graph is disconnected. It repeats the second step for a certain number of times, and finally returns the knockout set with smallest size. However, the problem formalized in [74] does not consider the impact of the knockout to the other parts of the network. For example, it only aims to find a minimal set of nodes that disconnect, say, set T from set S, but it does not take into account minimizing the number of "non-T" parts disconnected from S.

Some theoretical work focuses on finding *all* minimal separators of graphs [7, 52]. Berry *et al.* [7] proposed an algorithm to obtain all minimal 2-separators with time complexity $O(n^3)$ per separator. However, it is not clear how to apply these methods to solve constrained minimal separator problems efficiently. In general, it is infeasible to list all minimal separators and then test them one-by-one.

Correspondences between lattices (as used in itemset mining [69,71,88]) and graph separators have been explored [9]. It is proved that, given a binary relation, there is a one-to-one map between the elements of the itemset concept lattice and minimal separators of an undirected graph [8]. The underlying undirected graph is a co-bipartite graph (i.e., the complement of a bipartite graph) where one part represents the set of objects (e.g., genes) and the other part is the set of properties (e.g., sub-systems or biological processes). For each concept, the complement of the union of its intent and extent with respect to the whole sets of objects and properties corresponds to a minimal separator of the graph.

We present an algorithm to efficiently compute constrained minimal separators. First, it models the relations between genes (objects) and processes (labels) as a bipartite graph. Given two sets of processes to be separated, it then quickly enumerates all subgraphs that contain the potential min-

imal separators by harnessing the closure operator of the corresponding concept lattice. Second, the minimal separators of each set of processes that have to be excluded from the final results are computed as filters for the next step, using a newly developed efficient method. Third, the minimal separators of each candidate subgraph are generated, using the output of the second step as filters. We prove the correctness of the algorithm theoretically and experimentally.

3.3 Problem Definition

The notion of a graph minimal separator was first introduced by Dirac [19], and has been extensively studied [7, 73]. The definition of minimal separator is given below, please refer to [29].

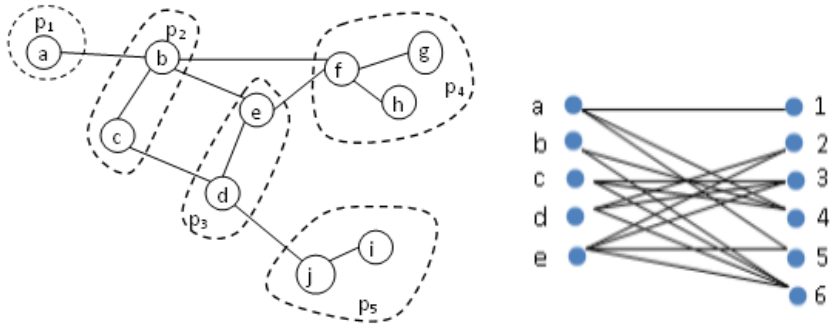
Definition 3.3.1. Given a graph $G(V, E)$ and two non adjacent vertices a and b, a subset $S \subset V$ is an (a, b)-separator if the removal of S separates a and b into different connected components. If no proper subset of S is an (a, b)-separator, then S is referred to as a minimal (a, b)-separator. A *minimal separator* is a set of vertices S for which there exist non-adjacent vertices a and b such that S is a minimal (a, b)-separator.

Let $\mathcal{O} = \{o_1, o_2, ..., o_n\}$ be a set of objects. Objects interact with each other, and these interactions are encoded in an undirected graph $G_{\mathcal{O}}$, where vertices denote objects and edges denote interactions. Let $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ be a set of properties. Each object is associated with one or more properties. In this study, we do not consider the case when there are some objects that do not associate with any properties. The relationship between objects and properties is captured in a relation $E \subseteq \mathcal{P} \times \mathcal{O}$, where for any $o \in \mathcal{O}$ and any $p \in \mathcal{P}$, there is $(o, p) \in E$, if the object o has the property p, otherwise $(o, p) \notin E$. The whole network is represented as a bipartite graph $\mathcal{G}(\mathcal{P}, \mathcal{O}, E)$. Each property p_i corresponds to a subgraph of $G_{\mathcal{O}}$, $G_{\mathcal{O}}(\{p_i\})$, that is induced by the set of objects associated with p_i, and edges between them. Two properties p_i and p_j are said to be *connected* if their corresponding subgraphs share objects or objects associated with two properties are connected in the object network; otherwise they are said to be *disconnected*.

Suppose that two sets of properties $P \subset \mathcal{P}$ and $Q \subset \mathcal{P}$ are required to be separated. The objective of constrained minimal separator mining is to find a minimal set of objects to delete from $G_{\mathcal{O}}$ such that P and Q are disconnected from each other, while the connectivities of objects in P and Q are retained as much as possible. Here, connectivity is measured in terms of the number of edges, so in other words, the separators should retain as many edges in each partition as possible.

Figure 3.1 (left) depicts an example where $\mathcal{O} = \{a, b, c, d, e, f, g, h, i, j\}$ and $\mathcal{P} = \{p_1, p_2, p_3, p_4, p_5\}$. If the goal is to separate p_1 from p_5, then two minimal separators are $\{b\}$ and $\{d\}$. However, $\{d\}$ has fewer connections with the other objects than $\{b\}$, hence this would be chosen as the solution to the constrained minimal separator problem.

Given an undirected graph, the problem of finding all minimal separators has been well studied. Many efficient algorithms with polynomial time complexity per separator have been proposed.



	a	b	c	d	e
1		x	x	x	x
2	x	x	x		
3	x	x			
4				x	x
5		x	x	x	
6	x				

Figure 3.1: (left) An example of an interaction graph between genes with labels on sub-networks denoting sub-processes. (middle) A co-bipartite graph $G(\mathcal{O}, \mathcal{P}, E)$. For simplicity the edges within O and within P are not shown. (right) The corresponding relation R denoting the complement of E.

Notice that, in general, the number of minimal separators can be exponential. As stated earlier, it has been proved that deciding whether there is a separator of cardinality less than a given integer is NP-hard [74]. Therefore, exhaustively listing all the minimal separators and then finding minimal separator that cause minimal influences to the separated components is practically infeasible. Generally, a gene network contains hundreds to thousands of genes and an order of magnitude greater number of interactions. It is thus important to develop an algorithm to quickly restrict the search space, i.e., to rapidly locate the subgraph that contains the real solution and has dramatically smaller size than the original graph. In this study, we do precisely this, by exploiting the correspondence between concept lattices and graph minimal separators.

3.4 Concept Lattices and Minimal Separators of Graphs

Definition 3.4.1. Given a set of objects $\mathcal{O}$, a set of properties $\mathcal{P}$, and a relation R that is defined as a subset of the Cartesian product $\mathcal{P} \times \mathcal{O}$, the triple $C = (\mathcal{P}, \mathcal{O}, R)$ is called a *context*. A *concept*, also called a *closed set*, is a sub-product $A \times B \subseteq R$ such that for every property $x \in A$ and every object $y \in B$, $(x, y) \in R$, and for all property $x' \in \mathcal{P} - A$, there exists an object $y \in B$, such that $(x', y) \notin R$, and for all objects $y' \in \mathcal{O} - B$, there exists a property $x \in A$ such that $(x, y') \notin R$. A is referred to as the *intent* of the concept, and B is referred to as its *extent*. The concepts, ordered by inclusion of the intents, define a *concept lattice*.

Definition 3.4.2. An undirected graph $G(V_1, V_2, E)$ is called a *co-bipartite graph* if it is the complement of a bipartite graph. In other words, G has two disjoint vertex sets V_1 and V_2, and E includes edges that constitute a clique of V_1, a clique of V_2, and possibly some edges between V_1 and V_2.

Berry and Sigyret [8] have identified a one-to-one correspondence between the concepts of a

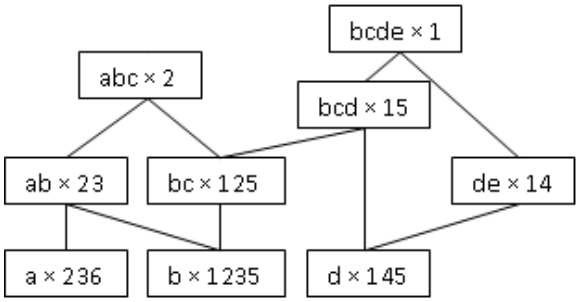


Figure 3.2: Example concept lattice, each element of which corresponds to a minimal separator.

concept lattice and the minimal separators of a co-bipartite graph. Given a co-bipartite graph $G(\mathcal{P}, \mathcal{O}, E)$, the mapping concept lattice is defined on the context $C(\mathcal{P}, \mathcal{O}, R)$ as follows:

1. $\mathcal{O}$ is an object set;
2. $\mathcal{P}$ is a property set;
3. $R \subseteq \mathcal{P} \times \mathcal{O}$ is a relation, for any $x \in \mathcal{P}$ and $y \in \mathcal{O}$, $(x, y) \in R$ if and only if $(x, y) \notin E$.

The last constraint says that the relation R is a complement of the relation represented by E. Suppose that a pair (A, B) is a concept of the context C, with $A \neq \emptyset$, $B \neq \emptyset$, $A \neq \mathcal{P}$ and $B \neq \mathcal{O}$. Then $S = (\mathcal{O} - B) \cup (\mathcal{P} - A)$ is a minimal separator of the graph G, which separates A from B.

Figure 3.1 (middle) shows an example of a co-bipartite graph between a set of properties $\mathcal{P} = \{a, b, c, d, e\}$ and a set of objects $\mathcal{O} = \{1, 2, 3, 4, 5, 6\}$. Edges between properties and between objects are not shown for simplicity. The corresponding relation R is shown in Figure 3.1 (right). The concept lattice built on R is shown in Figure 3.4. For instance, the concept $a, b \times 2, 3$ maps to a minimal separator $\{c, d, e, 1, 4, 5, 6\}$: namely, the one that separates $\{a, b\}$ from $\{2, 3\}$.

However, the applicability of the above correspondence is limited. The network (G) is required to be co-bipartite, whereas, in practice, most networks do not have this property. In this thesis, these requirements are loosened to a more general case where (a) the property set $\mathcal{P}$ is an independent set, i.e., no edge exists between any two properties; (b) the object set $\mathcal{O}$ is an arbitrary undirected graph; (c) there are edges between $\mathcal{P}$ and $\mathcal{O}$. This situation corresponds to the context of the problem defined earlier in Section 3.3. Here, $G_{\mathcal{O}}$ is the undirected graph of the object set and E is the set of edges between $\mathcal{O}$ and $\mathcal{P}$.

In the next section, we construct the mapping between a concept (or a closed set) and a constrained minimal separator and theoretically prove it. Figure 3.4 demonstrates the key steps of the construction and proofs. First, we establish the correspondence from a concept to a minimal separator that separates a set of properties from the rest but retains as much as possible the connectivities

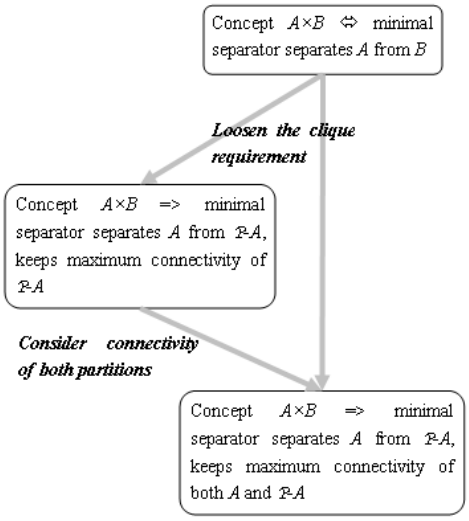


Figure 3.3: Steps of construction and proofs.

of the remaining properties. Second, based on the previous conclusion and the proof of the first step, we construct a minimal separator from concepts. This minimal separator disconnects a set of properties from the remaining properties and retains as much as possible the connectivities of both separated parts.

Some terms used in the rest of this chapter are defined below.

Definition 3.4.3. Given an undirected graph $G(V, E)$, if $X \subset V$, the *neighborhood* of X in G, denoted by $N_G(X)$, is the set of vertices that link to vertices in X, that is $N(X) = \{y \mid (x, y) \in E, x \in X, y \notin X\}$.

Definition 3.4.4. Given a graph $G(V, E)$ and a vertex set $A \subset V$, the *neighborhood graph of* A, denoted as $G(N(A))$, is defined as the subgraph of G that is composed of $N_G(A)$ and the edges between vertices in $N_G(A)$.

Definition 3.4.5. A separator S is called a *k-minsep* of a graph $G(V, E)$, if there are k vertices $\{a_1, a_2, ..., a_k\}$, with $k \geq 2$, such that each vertex a_i is in a different component of $G(V - S)$ and no proper subset of S has this property.

, separating vertices $\{a_1, a_2, ..., a_k\}$ into k different connected components, is

Notice that every k-minsep ($k \geq 3$) is a union of 2-minseps. For example, a k-minsep, which separating vertices $\{a_1, a_2, ..., a_k\}$ into k different connected components, is a combination of $a_i a_j$-minimal separators where $i \neq j, i, j \leq k$. Since for every pair of connected components, C_{a_i} and C_{a_j}, there is a set of vertices in the separator that link to both components, then this set will form an $a_i a_j$-minimal separator. Obviously, there is no vertex in the separator that links to only one component, otherwise the separator will not be minimal.

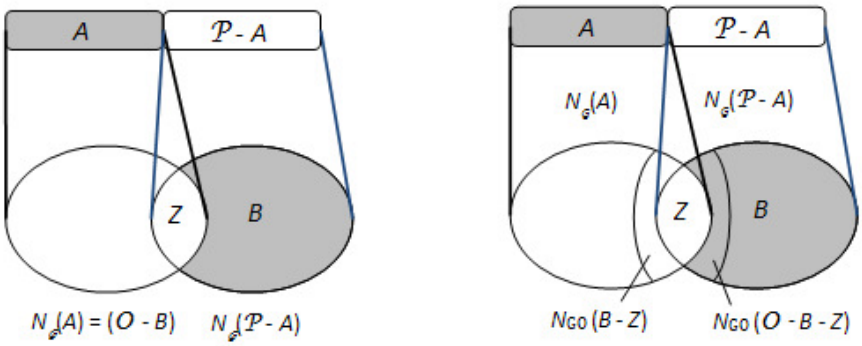


Figure 3.4: (left) Relationship between A and $\mathcal{P} - A$ described in Lemma 3.4.6. (right) Relationship between $N_{G_\mathcal{O}}(B) - Z$ and $N_{G_\mathcal{O}}(O - B - Z)$ described in Lemma 3.4.7.

3.4.1 Mining Constrained Minimal Separators

Given a graph and its mapping context as defined in the previous section, Lemma 3.4.6 below gives the mapping between an element of the concept lattice and the minimal separator that separates a set of properties from the rest and yet retains the connectivities in one of the disconnected parts as much as possible, among all minimal separators.

Lemma 3.4.6. *Let $\mathcal{G}(\mathcal{P}, \mathcal{O}, E)$ be a bipartite graph and $C = (\mathcal{P}, \mathcal{O}, R)$ be the corresponding context. Let $G_\mathcal{O}$ be the network of objects $\mathcal{O}$. Suppose that each object $o \in \mathcal{O}$ links to at least one property $p \in \mathcal{P}$ in $\mathcal{G}$. Let $A \times B$ be a concept in the context C. Then $S = N_{G_\mathcal{O}}(B) \cup (N_\mathcal{G}(\mathcal{P} - A) \cap N_\mathcal{G}(A))$ is a minimal separator that disconnects A from $\mathcal{P} - A$ and retains the connectivities of the subgraph induced by $\mathcal{P} - A$ as much as possible.*

Proof. The proof is structured as follows:

1. First, we prove that $N_\mathcal{G}(A)$ is a separator that separates A from $\mathcal{P} - A$. Since $\mathcal{P}$ is an independent set, two elements of $\mathcal{P}$ will be connected only indirectly, through objects in $\mathcal{O}$. $A \times B$ is a concept, so for all $a \in A$, for all $b \in B$, $(a, b) \in R$, i.e., $(a, b) \notin E$. Because every object links to at least one property, $N_\mathcal{G}(A) = \mathcal{O} - B$. This means that there is no path from A to $\mathcal{P} - A$ after removal of $\mathcal{O} - B$. Figure 3.4 (left) shows these relations. Therefore, $N_\mathcal{G}(A)$ is a separator that disconnects A from $\mathcal{P} - A$.

2. Second, based on the proof of the first step, $N_\mathcal{G}(A)$ must contain a minimal separator that disconnects A from $\mathcal{P} - A$. Let $S = N_{G_\mathcal{O}}(B) \cup (N_\mathcal{G}(\mathcal{P} - A) \cap N_\mathcal{G}(A))$. S is a subset of $N_\mathcal{G}(A)$, because $N_{G_\mathcal{O}}(B)$ is a subset of $\mathcal{O} - B$, and $N_\mathcal{G}(A) = \mathcal{O} - B$. Now, we prove that $S \subseteq N_\mathcal{G}(A)$ is a minimal separator that disconnects A from $\mathcal{P} - A$. Let $Z = N_\mathcal{G}(\mathcal{P} - A) \cap N_\mathcal{G}(A)$ be the neighbors of both A and $\mathcal{P} - A$ as shown in Figure 3.4(left), then S can be re-written as $S = N_{G_\mathcal{O}}(B) \cup Z$.

Suppose that S is not a separator, then there is an object $a \notin S$ but $a \in \mathcal{O} - B$, such that a links to A and has a path to $\mathcal{P} - A$ in the remaining graph. Thus, a either belongs to Z or has a path to an object in $N_{G_\mathcal{O}}$. This contradicts the definition of S. Suppose that S is not minimal, then there is a subset $S' \subset S$ that is also a separator. Let $a \in S$ but $a \notin S'$, then a should not belong to Z, otherwise S' will not be a separator. If $a \in N_{G_\mathcal{O}}(B)$, after removing S' from the graph $\mathcal{G}$, there still exists a path from A to $\mathcal{P} - A$. This contradicts that S' is a separator. Therefore, S is a minimal separator that disconnects A from $\mathcal{P} - A$.

3. Suppose that an object x does not associated with any properties. And x connects to two objects a and b. If a and b are neighbors of A and $\mathcal{P} - A$ respectively, and $a, b \notin S$, then S will not be a separator. Therefore, the constraint that every object is associated with one or more property is necessary.

4. Now, we prove that the minimal separator S has the smallest impact on the connectivities of the subgraph induced by $\mathcal{P} - A$, $G_\mathcal{O}(\mathcal{P} - A)$. Suppose that this is not true, then there must be another minimal separator S', $S' \nsubseteq \mathcal{O} - B$, such that S' separates A from $\mathcal{P} - A$ and results in better connectivities in $G_\mathcal{O}(\mathcal{P} - A)$ than S does. There should be a vertex $b \in S' \cap B$, because Z links to properties in both A and $\mathcal{P} - A$. Hence, S' must contain Z. The overlap between S and $N_\mathcal{G}(\mathcal{P} - A)$ is Z, but S' shares Z and b with $N_\mathcal{G}(\mathcal{P} - A)$. This contradicts the assumption about the minimal impact of the removal of S'.

□

Using the "closed" property of a concept, it is easy to answer whether the required minimal separators exist. For example, if a property p_1 needs to be separated from p_2 and p_3, but every concept containing p_1 also contains p_2, then this reveals that there could be no such separator. The minimal separator constructed in Lemma 3.4.6 is contained in the neighborhood of A. The connectivity of the graph induced by $\mathcal{P} - A$ remains at most, but the connectivity of A itself is not considered. Lemma 3.4.7 shows that how the minimal separator, which separates two sets of properties and remains the connectivities of both sets as much as possible, can be constructed from the corresponding concept lattice.

Let $\mathcal{G}(\mathcal{P}, \mathcal{O}, E)$ be a given bipartite graph, $G_\mathcal{O}$ be the graph of objects $\mathcal{O}$, and $C = (\mathcal{P}, \mathcal{O}, R)$ be the corresponding context. For a concept $A \times B$, let $Z = N_\mathcal{G}(A) \cap N_\mathcal{G}(\mathcal{P} - A)$ as shown in Figure 3.4 (left). We define a co-bipartite graph $G_{A\times B}(V_1, V_2, E')$ as follows:

- $V_1 = N_{G_\mathcal{O}}(B) - Z$, objects in V_1 form a clique;
- $V_2 = N_{G_\mathcal{O}}(\mathcal{O} - B - Z)$, objects in V_2 form a clique;
- for any pair of vertices $x \in V_1$, $y \in V_2$, $(x, y) \in E'$ if and only if there is a link between x and y in the graph $G_\mathcal{O}$.

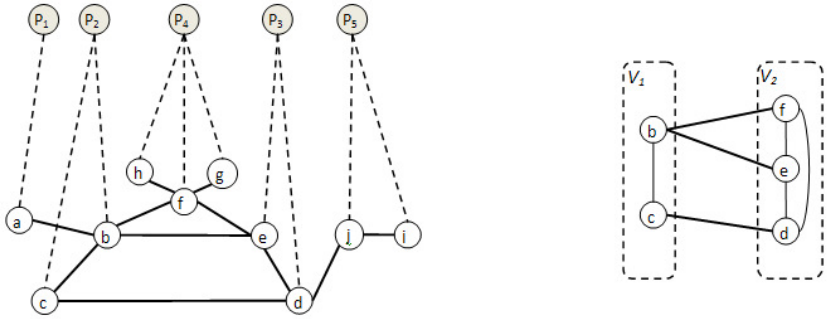


Figure 3.5: (left) The entire network of the example in Figure 3.1. (right) A co-bipartite graph defined to a concept $\{p_1, p_2\} \times \{d, e, f, g, h, i, j\}$.

The example shown in Figure 3.1(left) is re-drawn into a bipartite structure as depicted in Figure 3.5 (left). $\{p_1, p_2\} \times \{d, e, f, g, h, i, j\}$ is a corresponding concept. Figure 3.5 (right) shows the co-bipartite graph of this concept as defined above. Z is an empty set, because no objects associate with multiple properties. The co-bipartite graph has $V_1 = \{b, c\}$, $V_2 = \{e, d, f\}$, and $E' = \{(b, f), (b, e), (c, d))\}$.

Lemma 3.4.7. *Let $\mathcal{G}(\mathcal{P}, \mathcal{O}, E)$ be a given bipartite graph, $G_{\mathcal{O}}$ be the graph of objects $\mathcal{O}$, and $C = (\mathcal{P}, \mathcal{O}, R)$ be the corresponding context. Suppose that each object $o \in \mathcal{O}$ links to at least one property $p \in \mathcal{P}$. For every concept $A \times B$, let $G(V_1, V_2, E')$ denote the corresponding co-bipartite graph as defined above. Then, $S = S_G \cup (N_{\mathcal{G}}(\mathcal{P} - A) \cap N_{\mathcal{G}}(A))$ is a minimal separator that separates A from $\mathcal{P} - A$ and keeps graphs induced from both A and $\mathcal{P} - A$ as connected as possible internally, where S_G is a minimal separator that separates G into two connected components.*

Proof. The proof is structured as follows:

1. Lemma 3.4.6 proves that a subset of $\mathcal{O} - B$, $S' = N_{G_{\mathcal{O}}}(B) \cup (N_{\mathcal{G}}(\mathcal{P} - A) \cap N_{\mathcal{G}}(A))$ is a minimal separator that disconnects A from $\mathcal{P} - A$ while keeping the graph of $\mathcal{P} - A$ as connected as possible. Now, in addition to the connectivity of the graph of $\mathcal{P} - A$, the connectivity of the graph A also needs to be considered. A connects to $\mathcal{P} - A$ only through objects in $\mathcal{O}$, according to the definition of $\mathcal{G}$. The neighborhood of A,$N_{\mathcal{G}}(A)$, may overlap with $N_{\mathcal{G}}(\mathcal{P} - A)$ and/or some objects in $N_{\mathcal{G}}(A)$ may be adjacent to some objects in $N_{\mathcal{G}}(\mathcal{P} - A)$. Therefore, the minimal separator S must contain the overlaps between $N_{\mathcal{G}}(A)$ and $N_{\mathcal{G}}(\mathcal{P} - A)$ and some nodes that connect the two sets located in the boundary of two sets. Lemma 3.4.6 only considers the vertices located in the boundary of $N_{\mathcal{G}}(A)$.

2. Let $Z = N_{\mathcal{G}}(\mathcal{P} - A) \cap N_{\mathcal{G}}(A)$ be the overlaps between neighborhoods of A and $\mathcal{P} - A$. Since every object links to one or more properties, the neighborhood of $\mathcal{P} - A$ is the union of B and Z. Let $\mathcal{G}(-Z)$ denote the graph $\mathcal{G}$ after removing the vertices in Z. In $\mathcal{G}(-Z)$, $N_{G_{\mathcal{O}}}(B) - Z$ indicates the set of vertices in $\mathcal{O} - B$ that links to $N_{\mathcal{G}}(\mathcal{P} - A)$; and $N_{G_{\mathcal{O}}}(\mathcal{O} - B - Z)$ is the

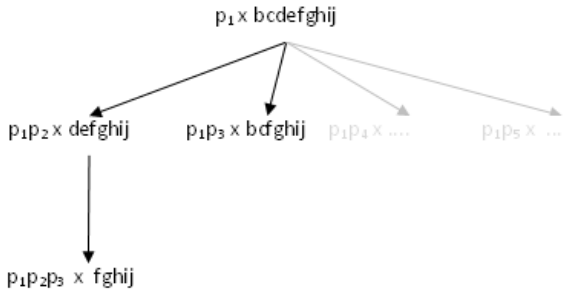


Figure 3.6: CHARM search and extend process.

set of vertices in B that directly links to $\mathcal{O} - B$ (i.e., $N_{\mathcal{G}}(A)$), as shown in Figure 3.4 (right). $N_{G_{\mathcal{O}}}(B) - Z$ and $N_{G_{\mathcal{O}}}(\mathcal{O} - B - Z)$ indicate the objects in the boundary that connect two induced graphs of A and $\mathcal{P} - A$.

3. $G(V_1, V_2, E')$ is the co-bipartite graph of $A \times B$. We can define a corresponding context $C'(V_1, V_2, R'_E)$, where R'_E is the complement of E'. Based on the previous work by Berry *et al.*, there is a one-to-one mapping between every concept generated on R'_E and every minimal separator that separates $G(V_1, V_2, E')$ into two connected components. Therefore, all the concepts generated on R'_E represent all the minimal separators that disconnect A from $\mathcal{P} - A$ in the original graph $\mathcal{G}$ after the removal of Z. Suppose that S_G is the minimal separator of $G(V_1, V_2, E')$ that results in minimal influences to connectivities of both graphs induced by A and $\mathcal{P} - A$. $S = S_G \cup Z$ is the minimal separator of $\mathcal{G}$ that disconnects A from $\mathcal{P} - A$ and keeps both parts as connected as possible.

□

3.5 Mining Constrained Minimal Separators

Lemma 3.4.7 provides a guide to mine the constrained minimal separators. It points out that the most important part is to find the minimal separator S_G of the co-bipartite graph $G(V_1, V_2, E')$ of a concept. The concept implicates the two property sets that are required to be disconnected. Obviously, the search space is reduced dramatically as the size of $G(V_1, V_2, E')$ is much less than the size of the original whole graph $\mathcal{G}$. However, it is not efficient to generate all of the minimal separators and test them one-by-one, since in some cases, the size of the co-bipartite graph may still be large. We propose an algorithm to efficiently generate S_G without listing all the minimal separators in Section 3.5.1. One issue is that S_G should not disconnect both separated components. Another issue is that S_G may not be unique. For instance, there may be two minimal separators S_{G1} and S_{G2}. Both of them keep the graphs induced by A and $\mathcal{P} - A$ connected respectively. In this case, the one that keeps more edges in the remaining graphs should be preferentially selected as the final answer.

Algoritmo 3.1 MinSepOfV$(G(V, E), Z)$

input :G is the graph with vertices V and edges E
input :Y is the set of vertices from which the
minimal separators are made
output: $\mathcal{F}$ the set of minimal separators

for $x \in Y$ **do**
$N'(x) \leftarrow Y \cap N(x)$;
for component $C \in G(V - \{x\} - N'(x)$ **do**
$\mathcal{F} \leftarrow \mathcal{F} \cup N(C)$;
end for
end for

$tmp \leftarrow \emptyset$;
while $\mathcal{F} - tmp \neq \emptyset$ **do**
$F \in \mathcal{F} - tmp$;
for $y \in F$ **do**
$N'(y) \leftarrow Z \cap N(y)$;
$\mathcal{F} \leftarrow \mathcal{F} \cup \{N(C) | C \in G(V - F - N'(y))\}$;
end for
$tmp \leftarrow tmp \cup \{F\}$;
end whilereturn $\mathcal{F}$.

3.5.1 Quickly Generate S_G

Let $G_{\mathcal{O}-Z}(A)$ and $G_{\mathcal{O}-Z}(\mathcal{P}-A)$ be the graphs induced by A and $\mathcal{P}-A$ respectively from $\mathcal{G}(-Z)$. Let $\mathcal{G}(-Z)$ be the graph after removing Z from $\mathcal{G}$. To retain the connectivities of both graphs, S_G should not contain any minimal separators of $G_{\mathcal{O}-Z}(A)$ or any minimal separators of $G_{\mathcal{O}-Z}(\mathcal{P}-A)$. Notice that only those minimal separators that are composed of vertices in $N_{G_\mathcal{O}}(B)-Z$, and those minimal separators that consist of vertices in $N_{G_\mathcal{O}}(\mathcal{O}-B-Z)$, are necessary to be used as filters. The question can be further simplified as to generate only 2-minseps from a given subset of vertices, since every k-minsep, $k \geq 3$, is a combination of several 2-minseps. Kloks and Kratsch [52] proposed an approach to use a minimal (a,b)-separator to generate another (a,b)-minimal separator. This method was extended to generate all 2-minseps with time complexity $O(n^3)$ per separator [7].

Given a graph $G(V,E)$ and a set of minimal separators X, it is known that if $x \in X$, for every component of $C \in G(V-X-N(x))$, the neighborhood of C is also a minimal separator [7]. Here, this approach is modified to generate minimal separators that are only composed of a specific set of vertices. We use $G_{\mathcal{O}-Z}(A)$ as an example to describe the idea of this method. Let V_A and E_A be the vertex set and the edge set. Let $Y = N_{G_\mathcal{O}}(B)-Z$ be the specific set of vertices in the boundary. For all vertices $y \in Y$, and every component $C \in G_{\mathcal{O}-Z}(A)(V_A-\{y\}-(N(y)\cap Y))$, $N(C)$ must be a minimal separator and $N(C)$ is a subset of Y. Applying the above theorem iteratively, if there is a minimal separator $S \subseteq Y$, for any vertex $x \in S$, and every component $C \in G_{\mathcal{O}-Z}(A)(V_A-S-(N(x)\cap Y))$, $N(C)$ is also a minimal separator and this separator is contained in Y. The details of this algorithm is described in Algorithm 3.1.

After these specific minimal separators of $G_{\mathcal{O}-Z}(A)$ and $G_{\mathcal{O}-Z}(\mathcal{P}-A)$, denoted by $\mathcal{F}$, are generated, they are used to guide the generation of minimal separators of the co-bipartite graph $G(N_{G_\mathcal{O}}(B)-Z$,$N_{G_\mathcal{O}}(\mathcal{O}-B-Z), E')$. The minimal separator S_G maps to a concept of the corresponding concept lattice, which has the following properties: neither the complement of the intent nor the complement of the extent contains any elements in $\mathcal{F}$.

3.5.2 Algorithm

The proposed algorithm is formally described in this section. In addition to the object network and the associations between properties and objects, the input also contains two sets of properties P and Q that need to be separated. The whole process has three steps. First, we compute the concepts on the relation between properties and objects whose intent contains P but not Q and store them in a list. When $P \cup Q$ is not equal to the whole set of properties $\mathcal{P}$, there can be multiple minimal separators depending on how the remaining properties are partitioned. Figure 3.6 shows an example. Suppose that the request is to separate p_1 from p_4 and p_5. There are multiple candidates, for instance, $\{p_1\} \times \{b,c,d,e,f,g,h,i,j\}$ and $\{p_1,p_2\} \times \{d,e,f,g,i,j\}$. We adapt the closed itemset mining algorithm CHARM [88] to compute the concepts where constraints are added in the process of extending the intent. Figure 3.6 demonstrates the search and extension process using the

Algoritmo 3.2 MinSepWithFilter($G(V_1, V_2), \mathcal{F}$)

input :G is the co-bipartite graph with two parts V_1 and V_2
input :$\mathcal{F}$ is a collection of the set of vertices that the resulting minimal separators should not contain
output: $\mathcal{S}$ the set of minimal separators

transform vertices of V_1 to transaction ids and vertices of V_2 as items
define transactions based on the negation of edges between V_1 and V_2,
i.e., if $a \in V_1$, $b \in V_2$ is not adjacent, then transaction a contains item b.
using CHARM to generate the concept lattice saved in L
for $A \times B \in L$ **do**
  $S \leftarrow (V_1 - A) \cup (V_2 - B)$;
  if $S \notin \mathcal{F}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{S\}$;
  end if
end for

return $\mathcal{S}$.

example data shown earlier. The intent containing p_1 is extended, but the properties p_4 and p_5 will not be included. The extension stops when no more properties can be extended. If every concept with the intent including A also includes B, then there are no such separators satisfying the separation request. Second, one concept $A \times B$ is picked from the list. The overlap between $N_{\mathcal{G}}(A)$ and $N_{\mathcal{G}}(\mathcal{P} - A)$ is computed and saved in Z. Then the minimal separators of the graph induced by A only contain vertices in $N_{G_{\mathcal{O}}}(B) - Z$, and the minimal separators of the graph induced by $\mathcal{P} - A$ that only contain vertices of $N_{G_{\mathcal{O}}}(\mathcal{O} - B - Z)$ are computed respectively. The detailed algorithm is described in Algorithm 3.1. Third, we compute the minimal separator of the co-bipartite graph that consists of $N_{G_{\mathcal{O}}}(\mathcal{O} - B - Z)$ and $N_{G_{\mathcal{O}}}(B) - Z$. The minimal separators generated in the second step serve as filters. The detailed approach is presented in Algorithm 3.2. The second and third steps will be repeated until all candidates are examined. If there are multiple results, then those Ss removing the minimal number of edges will be selected. The final result will be the union of S and Z.

3.6 Experiments

Dataset. The characteristics of the datasets used for experimental evaluation are shown in Table 4.1. The yeast pathways, including both metabolic pathways and regulatory pathways, are download from the Kyoto Encyclopedia of Genes and Genomes (KEGG) [48]. In this study, the goal is to find a minimal set of genes to knock out such that two sets of pathways are disconnected from each other and both sets are connected within themselves as much as possible. Therefore, only genes are considered; any other chemical compounds and directions of reactions are ignored.

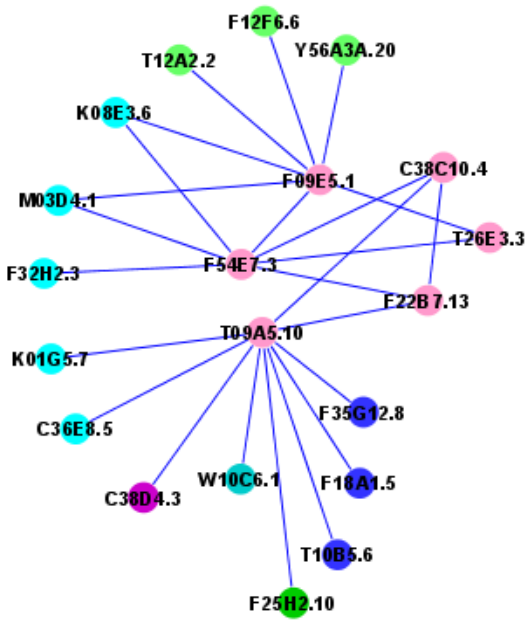


Figure 3.7: A portion of the *C. elegans* integrated gene network.

Table 3.1: Dataset characteristics.

Dataset	#objects	#interaction	#labels
S. cerevisiae interactions	1217	3877	105
C. elegans integrated gene network	259	4588	16

Two genes are connected if their products (enzymes) participate in successive reactions within the original pathway. Two pathways are connected if they share the same gene or the product of one pathway is the substrate of the other one. The *Caenorhabditis elegans* gene network is taken from the study of *C. elegans* early embryogenesis [35]. Each gene is assigned to one or more "molecular machines" in this study. The interactions of genes are extracted from the recently proposed gene network of *C. elegans* [56]. Here, we desire to identify the minimal set of genes to delete or knock down such that two sets of molecular machines are separated with each set remaining as connected as possible.

Yeast. In this experiment, we propose to separate the "Cell Cycle" pathway from "Carbohydrate Metabolism" and "Energy Metabolism". In the dataset, there are: one "Cell Cycle" pathway, 15 'Carbohydrate Metabolism" pathways, and 6 "Energy Metabolism" pathways. Our algorithm yields 5 genes as the result: *FUS3*, *FAR1*, *SWI4*, *SWI6*, and *PHO5*. All "Carbohydrate Metabolism" and "Energy Metabolism" pathways are kept the same as before when these 5 genes are removed. Only the "Cell Cycle" pathway is disconnected, but the connectivities of the rest of the pathways remain unchanged. All these 5 genes are located at the boundary of "Cell Cycle" pathway and the other pathways. In particular, the first 4 genes are located at the end of the "MAPK signaling"

pathway but the start of "Cell Cycle". Therefore, one can predict that the deletion of these genes should induce mostly cell cycle related phenotypes. Previous work shows that knockout of *SWI14*, *SWI6* cause cell cycle defects, budding mutants, and cell wall mutants [62,78]; the removal of *FAR1* induces mating defects and cell cycle arrest [83]; and *FUS3* is known to control the transition into cell cycle arrest [14]. *PHO5* is located at the end of the "Cell Cycle" pathway but at the start of "gamma-Hexachlorocyclohexane degradation" and "Riboflavin metabolism" pathways. It is known that *PHO5* causes nutrient relevant mutants. In our dataset, there are 109 genes involved in "Cell Cycle". The deletions of these 5 genes only slightly influence the connectivities of the entire pathway. It also has been proved biologically that the phenotypes resulting from the knockdown of the above genes are all viable [28]. Therefore, removing these 5 genes can separate the desired pathways and also keep both parts as connected as possible.

C. elegans. In this case study, "Cell Polarity" is separated from "Oocyte Integrity" and "Meiosis". "Cell Polarity" is connected to the other 6 molecular machines and "Oocyte Integrity" and "Meiosis" connect to all of them. 6 genes are picked for deletion based on our algorithm. Those are *T09A5.10*, *F09E5.1*, *F25H2.10*, *K08E3.6*, *M03D4.1*, and *F32H2.3*. The first two genes are categorized into "Cell Polarity", the third gene belongs to "Ribosome", and the remaining are contained in "Microtubule Cytoskeleton". Figure 4.12 shows the subgraph of "Cell Polarity" and its neighborhood, where colors represent different molecular machines. Pink nodes represent genes "Cell Polarity", among which three genes are connected to genes of other components. It is obvious that *T09A5.10* must be deleted, otherwise the other 7 genes have to be removed, which affects the connectivities of other components. Between *F09E5.1* and *F54E7.3*, *F09E5.1* is selected, because it connects with more outside genes. Although *F54E7.3* still connects with other components after removing the above two genes, it is retained, since the removal of this gene will cause the "Cell Polarity" sub-network to become disconnected. After removing these genes, the disconnected components are still connected within each other. For visualization simplicity, the connections between genes of remaining molecular machines are not shown in Figure 4.12.

3.7 Discussion

We have defined a new constrained minimal separator problem and designed an algorithm based on the relationships between concept lattices and minimal separators in graphs. Instead of considering the whole graph at one time, our algorithm quickly locates the subgraph where the actual minimal separators lie. A set of candidate subgraphs is generated by computing closures on relationships between labels and objects. The minimal separators of the candidate subgraphs are generated using the mappings between a concept lattice and graph minimal separators. A minimal separator of the candidate subgraph combined with the shared objects between two sets of labels forms the actual separator of these two sets. We proved the correctness of the proposed algorithm theoretically and evaluated its performance on biological datasets.

Chapter 4

Mining Biclusters with Supplied Partial Orders

In this chapter, we generalize the notion of biclusters over sets to biclusters over partial orders. We present formalisms and algorithms for mining such biclusters from binary datasets. This problem finds application in many structured data mining problems where prior domain-specific information about orders must be harnessed to yield meaningful clusters of entities. We have applied the algorithms for study the coupling of the co-regulation of genes in metabolic pathways in *E.coli*.

4.1 Introduction

Biclustering is a well studied technique to identify sub-groups of related entities in bipartite graphs. The input to bicluster mining [58] is a set of instances of a relationship between two domains. Figure 4.1 describes relationships between dates (rows) and weather conditions (columns) in Blacksburg, VA. A bicluster is a subset of rows along with a subset of columns with the property that each row element is related to each column element (stricter notions of biclusters are covered in [58] and later in our work). Figure 4.1 (right) lays out seven biclusters in the matrix as contiguous sub-matrices by re-ordering the rows and columns of the matrix [33], repeating rows and columns if necessary. For example, the bicluster spanning rows three through six and columns two through four states that each of the four days from July 1–4, 2004 experienced each of the weather conditions "$>$ 60 F," "Daylight $>$ 10 h," and "Cloudy."

While biclustering (and variations of it, such as co-clustering [17]) is a well studied topic, most existing algorithms assume that the domains (over which biclusters are mined) are unordered sets. However, in real applications, many domains have natural orderings that we would like biclusters to either obey or exploit. For example, genes are located in the DNA sequences along a total order, metabolic reactions are partially ordered according to DAGs of pathways, and so on. Figure 4.2 describes the high osmolarity glycerol pathway in *S.cerevisiae*. Glucose is consumed by the organ-

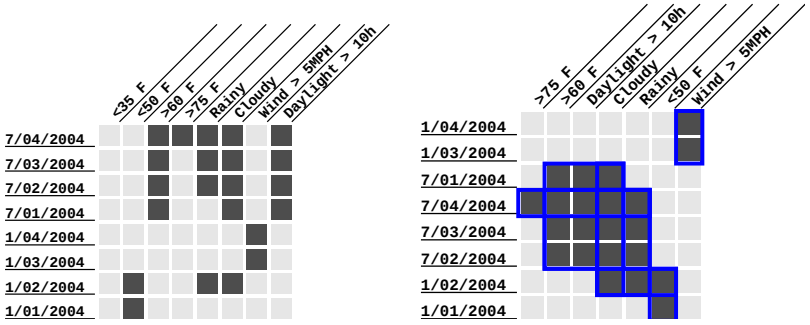


Figure 4.1: (left) Example input to biclustering. (right) Layout of computed biclusters.

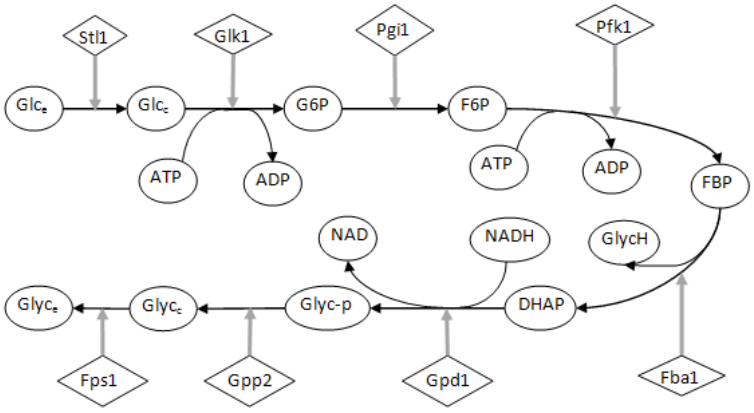


Figure 4.2: Osmolarity glycerol pathway in yeast.

ism and transformed to Glycerol and transported to the environment through a series of reactions. Each reaction is catalyzed by an enzyme. Consider the binary matrix of reactions and enzymes, and assume that the reactions are ordered as they appear in the pathway, with the enzymes also ordered accordingly. The resulting matrix is hence a diagonal matrix and a traditional biclustering algorithm would merely identify all the diagonal entries of the matrix individually as biclusters. A motivation for our work is to somehow 'push' the underlying order of reactions into the mining algorithm.

The exact problem as proposed above has not been studied before although there are close connections with some prior work. Mannila *et al.* [82] study the problem of finding a partial order of rows/columns from a binary matrix given an ordering over the other side (columns/rows). The principle behind their proposed algorithm is to find an ordering which induces as many consecutive ones as possible. A good application of this approach is in paleontological applications, where columns are species and rows are sites. The goal is to find the ordering of evolutionary stages of sites. The occurrences of species have natural partial ordering: species arise and disappear, and the existence of species thus forms an interval. The matrix captures the durations in which each species lives in the corresponding location. Therefore, when looking at each column, a good ordering of

sites should yield as many consecutive ones as possible.

4.2 Preliminaries

This section describes necessary background about partially ordered sets, biclusters and other notations used in subsequent sections. For a detailed introduction to these concepts, please see [16]. A *partial order* on a set O is a binary relation $\leq_P \subseteq O \times O$ that is reflexive, antisymmetric, and transitive, i.e., for all $a, b, c \in O$ the following properties hold:

1. *reflexive*: $(a, a) \in \leq_P$,
2. *antisymmetric*: $(a, b) \in \leq_P$ and $(b, a) \in \leq_P$ implies $a = b$, and
3. *transitive*: $(a, b) \in \leq_P$ and $(b, c) \in \leq_p$ implies $(a, c) \in \leq_P$.

The pair $(O, \leq_P)$ is referred to as a *partially ordered set* or a poset. For any $a, b \in O$, we write $a \leq_p b$ if $(a, b) \in \leq_P$. When the cardinality of O is finite, $(O, \leq_P)$ is referred to as a *finite poset*. A *strict* partial order on a set O is a binary relation that is *irreflexive* and *transitive*, written as $<_P$.

Given a poset $(O, \leq_P)$, two distinct elements $a, b \in O$ are referred to as *comparable*, written as $a \perp_P b$, if either $a \leq_P b$ or $b \leq_P a$; otherwise, a and b are $incomparable$, written as $a \parallel_P b$. If every two distinct elements $a, b \in O$ are comparable, then the poset $(O, \leq_P)$ is a *totally ordered set*, and the relation $\leq_P$ is a *total order* on O.

If $a \leq_P b$ and there is no $c \in O$ such that $a \leq_P c$ and $c \leq_P b$, then we say b *covers* a. A poset $(O, \leq_P)$ corresponds to a directed acyclic graph $G(O, E)$, where $E = \{(a, b) \mid b \text{ covers } a, \text{ and } a, b \in O\}$. This graph is called *Hasse diagram* of $(O, \leq_P)$, denoted by $H(O, \leq_P)$.

As an example, let $O = \{a, b, c, d, e, f, g\}$, and

$$\begin{aligned} \leq_P \;=\; & \{(a,b),(a,c),(a,d),(a,e),(a,f),(a,g), \\ & (b,c),(b,d),(b,e),(b,f),(b,g),(c,f)(c,g), \\ & (d,e),(d,f),(d,g),(e,f),(e,g),(f,g)\}. \end{aligned}$$

The corresponding Hasse diagram is shown in Figure 4.3. In this example, a and b are comparable (i.e., $a \perp_P b$) and, further, b covers a, but c and d are incomparable (i.e., $c \parallel_P d$).

A *linear extension* $\leq_P^*$ of a partial order $\leq_P$ on a set O is a total order, such that for any $a, b \in O$, $a \leq_P b$, it is also true that $a \leq_P^* b$. A partial order may have multiple linear extensions. For example, a linear extension of the above poset is $a \leq_p b \leq_p c \leq_P d \leq_p e \leq_P f \leq_P g$, and $a \leq_p b \leq_p d \leq_P c \leq_p e \leq_P f \leq_P g$ is another linear extension. If the partial order is an empty relation, then every possible total order of O is considered as a linear extension. For example, if the size of O is n, then the number of linear extensions is $n!$.

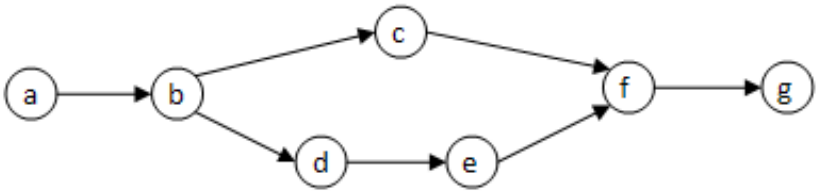


Figure 4.3: Hasse diagram of an example poset.

C is a *chain* of $(O, \leq_P)$, if $C \subseteq O$ and for every pair of distinct elements $a, b \in C$ is comparable, $a \perp_P b$. If $(O, \leq_P)$ is a totally ordered set, then O itself is a chain. An *antichain* of $(O, \leq_P)$ is a subset C of O in which every two distinct elements $a, b \in C$ are incomparable, i.e., $a \parallel_P b$. In the previous example, $C = \{a, b, e, f, g\}$ is a chain, and $C' = \{c, d\}$ is an antichain.

We define that a subset $C \subseteq O$ is a *consecutive chain* of $(O, \leq_P)$, when the subgraph $G'(C, \{(a, b) \in E \mid a \in C, b \in C\})$ of the Hasse diagram $G(O, E)$ forms a connected path. In other words, C is a chain, and in its linear extension $a_1 \leq_P a_2 \leq_P ... \leq_P a_{|C|}$, a_{i+1} covers a_i, $1 \leq i \leq |C| - 1$. Taking the above example, $C = \{a, b, f, g\}$ is a chain but not a consecutive chain. $C' = \{a, b, c, f, g\}$ is a consecutive chain, since it corresponds to a path $a \leq_P b \leq_P c \leq_P f \leq_P g$.

Given a poset $(O, \leq_P)$, let $\mathcal{P}(O)$ denote the power set of O. A mapping $\theta : \mathcal{P}(O) \longrightarrow \mathcal{P}(O)$ is a *closure operator* [25, 89] on O, if θ satisfies the following conditions for any $X, Y \subseteq O$:

(i) Extension: $X \subseteq \theta(X)$,

(ii) Monotonicity: $X \subseteq Y$ implies that $\theta(X) \subseteq \theta(Y)$, and

(iii) Idempotency: $\theta(\theta(X)) = \theta(X)$.

Given a binary matrix M and a total order $\leq_{T_r}$ on the row set, assume that the rows are ordered according to $\leq_{T_r}$. The *Lazarus count of columns* with respect to $\leq_{T_r}$, $L_c(M, \leq_{T_r})$, is defined as the sum of the number of zeros between ones in each column. Such zeros are called *Lazarus events*. Similarly, when a total order over the columns $\leq_{T_c}$ is given and the columns are ordered accordingly, the *Lazarus count of rows* with respect to the total order, $L_r(M, \leq_{T_c})$, is the sum of the number of zeros between ones in each row.

For example, in Figure 4.4(a), the Lazarus count of columns is 1, because in the third column, the third row has value 0. The Lazarus count of rows of the example shown in Figure 4.4(b) is also 1. The third example, Figure 4.4(c), has zero Lazarus count for both rows and columns.

The formal description of the old definition of a bicluster is given below. Let $R(A, B)$ be a relationship between entity sets A and B, i.e., $R \subseteq A \times B$. A *bicluster* (X, Y) *on* R is a set $X \subseteq A$ and a set $Y \subseteq B$ such that $X \times Y \subseteq R$, i.e., every pair of entities in $X \times Y$ belongs to R. Further, the bicluster (X, Y) is *closed* if

1:	0	1	1	0
2:	1	1	1	0
3:	0	1	0	0
4:	0	1	1	1

(a)

1:	0	1	1	0
2:	1	1	1	1
3:	0	1	1	1
4:	0	1	0	1

(b)

1:	1	1	1	1
2:	0	1	1	1
3:	0	1	1	0
4:	0	0	1	0

(c)

Figure 4.4: Example matrices.

(i) for every entity $x \in A - X$, there is some entity $y \in Y$ such that $(x, y) \notin R$, and

(ii) for every entity $y \in B - Y$, there is some entity $x \in X$ such that $(x, y) \notin R$.

That is, adding an entity in $A - X$ or $B - Y$ to the bicluster will violate the condition defining the bicluster.

4.3 Generalizing the Definition of Biclusters

In this section, we generalize the definition of biclusters to accommodate the notion of orders over the entity sets.

Let R be a binary relation between entity sets A and B, $R \subseteq A \times B$, and let $\leq_{P_A}$, $\leq_{P_B}$ be partial orders on A and B respectively. A *bicluster* (X, Y) on R is a set $X \subseteq A$ and a set $Y \subseteq B$ such that for all $x \in X$, $y \in Y$,

(i) for every linear extension $\leq^*_{P_A}$ of $\leq_{P_A}$, and for every linear extension $\leq^*_{P_B}$ of $\leq_{P_B}$, it is true that $L_r(R(X, Y), \leq^*_{P_B}) = 0$, and $L_c(R(X, Y), \leq^*_{P_A}) = 0$,

(ii) for all $x \in X$, $\sum_{y \in Y} R(x, y) \geq s$, and , and for any $y \in Y$, $\sum_{x \in X} R(x, y) \geq s$,

where s is a user-specified support threshold. When s is a positive number, the second condition will limit the case when an entire row or column is zeros.

The old definition of biclusters requires that *every* pair of entities in the bicluster must obey the binary relation. In other words, every entry of the corresponding binary matrix should have value 1. The original notion of biclusters is thus a special case of the new definition of biclusters. When both entity sets A and B are non-ordered sets, i.e., when $\leq_{P_A} = \emptyset$ and $\leq_{P_B} = \emptyset$, the new definition of biclusters will reduce to the original definition of biclusters.

The major difference between the newly defined bicluster and the old bicluster is that the new definition allows 0s in the binary matrix of the bicluster, when at least one of the entity sets has order information. More clearly, some entity pairs that do not obey the binary relation R are still allowed in the bicluster, as long as the the ordered entity subset contributes a zero Lazarus count.

Figure 4.5 illustrates the characteristics of the new definition of biclusters. When both rows and columns are non-ordered sets, the bicluster obeys the original definition as shown in Figure 4.5 (right). When one entity set is totally ordered and the other is non-ordered, the pattern shown in Figure 4.5 (left) is also a legal bicluster. When both entity sets are totally ordered, the pattern shown in Figure 4.5 (middle) is also allowed.

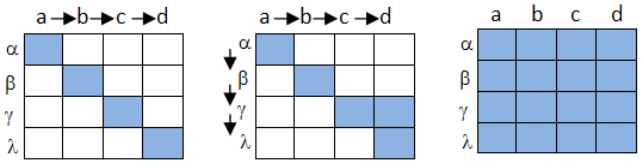


Figure 4.5: (left) Rows are non-ordered, and columns are totally ordered. (middle) Both rows and columns are totally ordered. (right) both are non-ordered.

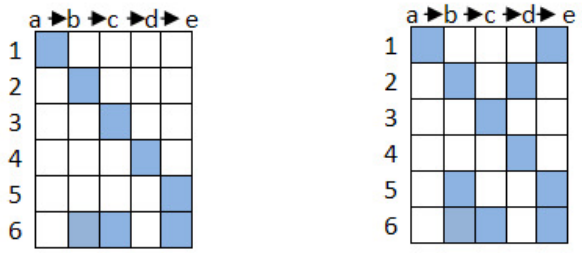


Figure 4.6: Examples of binary relations between two entity sets. Rows are non-ordered, and columns are totally ordered. (left) Relation 1. (right) Relation 2.

4.3.1 Closed Biclusters

Figure 4.6 (left) illustrates an example of a binary relation on two data sets $\{a, b, c, d, e\}$ and $\{1, 2, 3, 4, 5, 6\}$. Suppose that ϵ is 1, $(\{a, b, c, d\}, \{1, 2, 3\})$ is a bicluster, as is $(\{a, b, c, d, e\}, \{1, 2, 3, 4, 5\})$, and the latter contains the former. A bicluster that is not contained in any other proper bicluster is called a *closed* bicluster. A set may correspond to multiple closed biclusters. For example, given a set $\{1, 2, 3\}$ as shown in Figure 4.6 (right), both $(\{1, 2, 3\}, \{a, b, c\})$ and $(\{1, 2, 3\}, \{c, d, e\})$ are closed biclusters. In this section, a closure operator on an entity set is described, and a formal definition of a closed bicluster is given based on the closure operator.

Let $(A, \leq_{P_A})$ and $(B, \leq_{P_B})$ be two posets, and R be a binary relation on A and B, $R \subseteq A \times B$. Let $\mathcal{P}(A)$ and $\mathcal{P}(B)$ denote the power sets of A and B and consider $X \subseteq A$ and $Y \subseteq B$.

A pair (X, Y) is called a *valid* pair, if for any linear extensions $\leq^*_{P_A}, \leq^*_{P_B}$ of $\leq_{P_A}$ and $\leq_{P_B}$, the Lazarus counts of rows and columns of the matrix derived by X and Y are zeros, i.e., $L_c(R(X, Y), \leq^*_{P_A}$

$) = 0, L_r(R(X,Y), \leq^*_{P_B}) = 0$. In a valid pair (X,Y), Y is called *maximal*, if there is no proper superset $Y' \supset Y$, such that (X,Y') is also a valid pair. These definitions can be similarly posed over X. We define two relations as follows:

$$\phi \subseteq \mathcal{P}(A) \times \mathcal{P}(B),\ (X,K) \in \phi, \quad \text{if } K \subseteq B, (X,K) \text{ is a valid pair}, \forall x \in X, \sum_{y \in Y} R(x,y) \geq s, \text{ and in this pair,K is maximal.}$$

$$\varphi \subseteq \mathcal{P}(B) \times \mathcal{P}(A),\ (Y,I) \in \varphi, \quad \text{if } I \subseteq A, (Y,I) \text{ is a valid pair}, \forall y \in Y, \sum_{x \in I} R(x,y) \geq s, \text{ and in this pair,I is maximal.}$$

Lemma 4.3.1. *Let $\phi \circ \varphi$ denote the composition of two relations ϕ and φ, $\phi \circ \varphi \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ and dually let $\varphi \circ \phi$ denote the composition of φ and ϕ, $\varphi \circ \phi \subseteq \mathcal{P}(B) \times \mathcal{P}(B)$. Then both $\phi \circ \varphi$ and $\varphi \circ \phi$ are closure operators.*

Proof. Let $(A, \leq_A)$ and $(B, \leq_B)$ be two posets.

1. Extension: Let X be a subset of A. The extension property means that for all $X' \subseteq A$, $(X,X') \in \phi \circ \varphi$, $X \subseteq X'$ holds. Suppose that there is a subset X', $(X,X') \in \phi \circ \varphi$ and $X \not\subseteq X'$. Let $\mathcal{Y} = \{Y \mid (X,Y) \in \phi\}$, then there is $Y \in \mathcal{Y}$, and $(Y,X')\varphi$. This means that there is $x \in X$, $x \notin X'$, and based on the definition of φ, $L_r(R(x,Y), \leq^*_{P_B}) > 0$. This contradicts the definition of ϕ.

2. Monotonicity: Let X_1, X_2 be two subsets of A, and $X_1 \subseteq X_2$. The goal is to prove that for all $X_1' \subseteq A$, $(X_1, X_1') \in \phi \circ \varphi$ and for all $X_2' \subseteq A$, $(X_2, X_2') \in \phi \circ \varphi$, $X_1' \subseteq X_2'$. Let $(X_1, Y_1) \in \phi$, then based on the definition of ϕ, there is $(X_2, Y_2) \in \phi$ and $Y_2 \subseteq Y_1$. Let $(Y_2, X_2') \in \varphi$, then there is $(Y_1, X_1') \in \varphi$, and based on the definition of φ, $X_1' \subseteq X_2'$ holds.

3. Idempotency:

   Let X be a subset of A, and $(X,X') \in \phi \circ \varphi$, $(X',X'') \in \phi \circ \varphi$. Based on the proofs of the above two properties "Extension" and "Monotonicity", it is true that $X \subseteq X'$ and $X' \subseteq X''$. Suppose that $X' \subset X''$, then there is an element $x \in X''$ but $x \notin X'$. Let $(X',Y) \in \phi$ and $(Y,X'') \in \varphi$, then based on the definition of ϕ, there is $Y' \subseteq Y$, such that $(X,Y') \in \phi$. Therefore, x should be contained in X' also. This contradicts the previous assumption that $x \notin X'$.

□

For every subset $X \subseteq A$, if there is $X \subseteq X'$ and $(X,X') \in \phi \circ \varphi$, then X' is a closure of X, and X' is a closed set. Suppose that $Y \subseteq B$, $(X,Y) \in \phi$, then there is $X \subseteq X'$, $(Y,X') \in \varphi$, and (X',Y) is a closed bicluster. Considering the example shown in Figure 4.6 (left), $(\{1,2,3\}, \{a,b,c,d,e\}) \in \phi$ and $(\{1,2,3,4,5\}, \{a,b,c,d,e\}) \in \varphi$, therefore $\{1,2,3,4,5\}$ is a closed set and $(\{1,2,3,4,5\}, \{a,b,c,d,e\})$ is a closed bicluster.

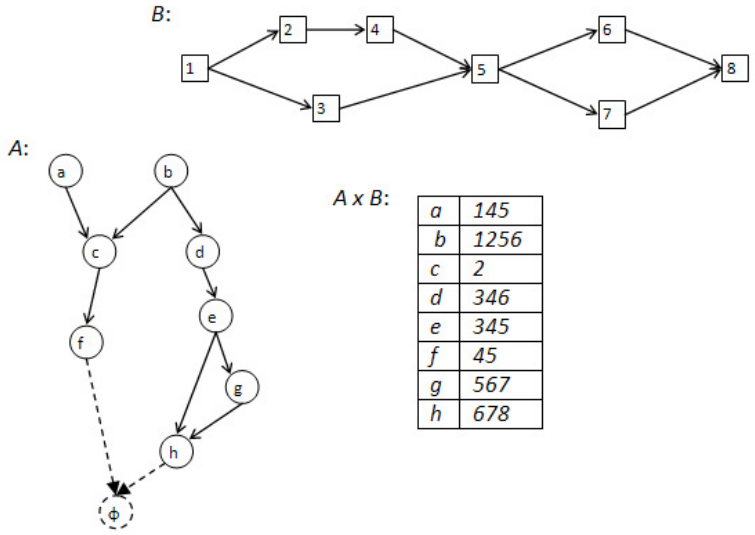
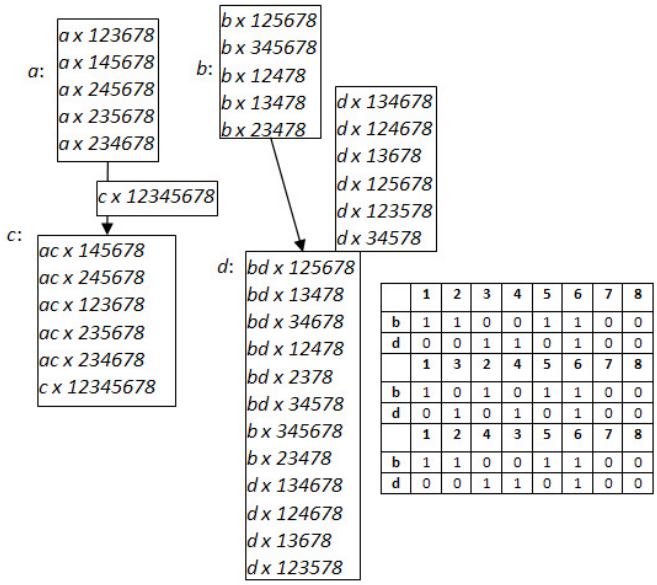


Figure 4.7: An example of two posets and a relation between them.

4.4 Algorithm

The algorithm for computing closed biclusters is described in this section. Assume we are given two posets $(A, \leq_{P_A})$ and $(B, \leq_{P_B})$. $X = a_1 a_2 ... a_n$ is a sequence of $(A, \leq_{P_A})$, written as $X \subseteq (A, \leq_{P_A})$, if $a_i \in A$ for $1 \leq i \leq n$ and X is consistent with a linear extension of $(A, \leq_{P_A})$. An AB_pair is a pair of sequences from the two sets A and B respectively, $X \subseteq (A, \leq_{P_A})$ and $Y \subseteq (B, \leq_{P_B})$, written as $X \times Y$. When the entity set is non-ordered, sequences with any order of entities are allowed. An AB_pair is considered as a candidate bicluster (or *valid*), if the Lazarus count of rows/columns of the binary matrix $R(X, Y)$ is zero. (Notice that when computing the Lazarus count, the matrices $R(X, Y)$ with all possible combinations of orders of rows and columns must be computed. All matrices must have zero Lazarus count.)

Before describing the algorithm, for ease of simplicity, we refer to a node in a Hasse diagram that does not have preceding nodes as a "root node", and a node that does not have following nodes as an "end node". Figure 4.7 shows two posets and a relation between them. The partial orders are described using Hasse diagrams. The whole procedure of computing closed biclusters is accomplished by going through the Hasse diagram level by level. Notice that when there is no partial order information for a given set, the Hasse diagram will reduce to a set of singletons. At the beginning, for each element in A, all AB_pairs are computed by searching the Hasse diagram of B. Notice that every AB_pair computed in this step only has one element in the A part. Next, the A part of the AB_pairs are extended starting from the root nodes of the Hasse diagram of A. As shown in Figure 4.8, there are two root nodes a and b with AB_pairs $a \times 145$ and $b \times 1256$. The child node (following element) of a is c. To extend the sequence a to ac, the two AB_pairs $a \times 145$ and $c \times 2$ need to be merged. The merged sequence of B parts is 1245. However, $ac \times 1245$ is not valid, because the Lazarus count of a is not zero. The sequence 1245 has to be separated



	1	2	3	4	5	6	7	8
b	1	1	0	0	1	1	0	0
d	0	0	1	1	0	1	0	0

	1	3	2	4	5	6	7	8
b	1	0	1	0	1	1	0	0
d	0	1	0	1	0	1	0	0

	1	2	4	3	5	6	7	8
b	1	1	0	0	1	1	0	0
d	0	0	1	1	0	1	0	0

Figure 4.8: Outline of procedure for computing biclusters, Step 1.

and the separated subsequences must be maximal. Therefore, the valid AB_pairs are $ac \times 245$ and $ac \times 12$. Because both of them do not contain $a \times 145$, $a \times 145$ should be saved for possible later extensions.

Similar rules apply to the root node b. d is a child node of b. The B-parts of $b \times 1256$ and $d \times 346$ have three possible combinations, viz. 123456, 132456, and 124356. The valid AB_pairs must have zero Lazarus count in the three possible orderings. A node (or element) is not to be processed until all its parents are visited, i.e., nodes of the Hasse diagram are accessed in a breadth-first search (BFS) order. For example, node c has another parent b, so before extending to the following nodes of c, the extension from b to c should be completed first. Similar to the previous steps, the valid AB_pairs of bc are saved in node c, and meanwhile the AB_pairs of bc should merge with previously computed AB_pairs of ac. The results are shown in Figure 4.9.

When there are multiple end nodes, a virtual end node will be added to the Hasse diagram, and all original end nodes will have an arrow to the virtual end node. The reason for adding a virtual end node is to calculate combinations of AB_pairs between different branches. For example, as shown in Figure 4.7, the actual end nodes of A are f and h, i is a virtual end node, and both f and h have an arrow to i respectively. In the case that the partial order information is not given, the combinations only happen when visiting the virtual end node. It is not necessary to check all linear extensions of the elements (when the size of the set is n, there will be $n!$ linear extensions). Because whenever there are two or more 1s, it is not possible to add any zeros.

Algorithm 4.1 formally describes our procedure for computing closed new biclusters. In the pre-processing step, the Hasse diagrams of both posets are created. When a partial order is an empty set, the original Hasse diagram is a set of singletons. In this case, a virtual end node will be added

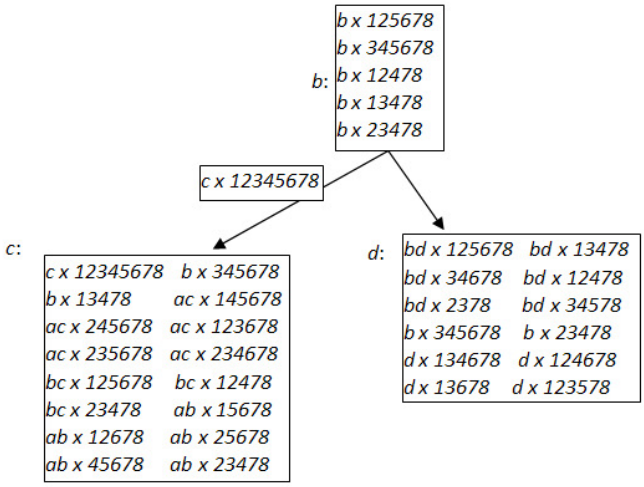


Figure 4.9: Outline of procedure for computing biclusters, Step 2.

in the Hasse diagram, and a set of arrows starting from every original node and pointing to the virtual node will be added. When a Hasse diagram has multiple end nodes, a virtual end node also will be added. Therefore, every Hasse diagram has only one end node. In the first step, for each element of A, the corresponding sequences of elements in B are computed. In other words, AB_pairs with single element in A and zero Lazarus counts are calculated. There may be multiple sequences of B matching an element of A. In the second step, the A parts of the AB_pairs calculated in the first step are extended. An AB_pair with zero Lazarus count and no further extensions on both A and B parts will be a closed bicluster. Algorithm 4.2 and Algorithm 4.3 demonstrate the first and the second step in details.

Algoritmo 4.1 Algorithm for computing biclusters over partial orders.

Input: Partially ordered set $(A, \leq_{P_A})$
Input: Partially ordered set $(B, \leq_{P_B})$
Input: Binary relation R on A and B
Output: Closed biclusters $\mathcal{C}$
`/* create Hasse diagrams of both posets */`
$H_B =$ Hasse digram of $(B, \leq_{P_B})$;
$H_A =$ Hasse diagram of $(A, \leq_{P_A})$;
$\mathcal{P} = \emptyset$; `/* store` AB_pairs `*/`
`/* initializing` AB_pairs `for each element of` A `*/`
foreach $x \in A$ **do**
  $\mathcal{Y} = ABPairs\_singleA(R(x, B), H_B)$;
  $\mathcal{P} = \mathcal{P} \cup x \times \mathcal{Y}$;
`/* computing closed biclusters */`
$\mathcal{C} = \emptyset$; `/* store a set of closed biclusters */`
$\mathcal{C} = Extension(H_A, \mathcal{P})$;
return $\mathcal{C}$;

Initially, the maximal AB_pairs of each element of A are computed by searching through the Hasse diagram of $(B, \leq_B)$, as shown in Algorithm 4.2. Here, a maximal AB_pair $x \times S$ means that S can not be expanded by adding any elements to the head or to the end, i.e., there is no element $b \in B, b \notin B$ such that $R(x, bS) = 0$ or $R(x, Sb) = 0$. Leading and ending zeros are allowed in S, which is for simplicity in the second step in which the A parts will be extended. Therefore, for each element of A, there may be multiple corresponding sequences. Each node in the Hasse diagram is described with five terms $n(id, children, NParent, \mathcal{S}, visited)$, where id denotes the element, $children$ is a collection of the links to the child nodes, $NParent$ indicates the number of parent nodes of n, $\mathcal{S}$ is a set of candidate AB_pairs, and $visited$ records visiting times of n. $visited$ is used to count if all parents of a node have been processed. $\mathcal{S}$ stores entries of the form $X \times \{Y\}$, where the left part is a sequence, and the right part is a set of sequences. For example, for an element $x \in A$, when processing a node $n \in H_B$ if there are two corresponding sequences Y_1 and Y_2, then $n.\mathcal{S} = x \times \{Y_1, Y_2\}$. The first three terms are initialized when the Hasse diagram is built, but the last two terms will be updated. A queue Q is used to store the nodes that need to be processed and to control the access of nodes in first-come-first-served order. Notice that a node will be put into the queue only when all its parent nodes have been visited, i.e., when $n.visited = n.NParent$. Each sequence S of n is extended by adding the child node to the end of the sequence. When the Lazarus count of the new vector $R(x, Sc.id)$ is not zero, there are two following steps. First, S needs to be saved in the child node c, because S may be extendable by adding the offspring of c. Second, some elements of S will be deleted such that the new sequence can expand to c. There are two ways of modification: one is taking out all zeros after ones; another is removing ones. When adding a new sequence to the node c, the new sequence should not be the subset or superset of any other sequences already in c. When c has multiple parents, two sequences may come from different parents. In this case, the two sequences may be able to be combined

into one long sequence. After the new sequence is computed, it is necessary to check if it can be combined with already computed sequences to form a longer sequence.

Algoritmo 4.2 $ABPairs\_singleA$: Computing AB_pairs of each element of A.

Input: An element in A, x
Input: Binary vector, $R(x, B)$
Input: Hasse diagram of $(B, \leq_B)$, H
Output: $\mathcal{Y}$ = candidate sequences of B
$\mathcal{Y} = \emptyset$; $Q = \emptyset$;
/* Put root nodes into the queue */
$Q \leftarrow$ root nodes;

while $Q \neq \emptyset$ **do**
  $n \leftarrow$ the first entry of Q;
  if $n.children = \emptyset$ **then** $\mathcal{Y} = \mathcal{Y} \cup n.\mathcal{S}$; **else**
    foreach $c \in n.children$ **do**
      $\mathcal{S} = \emptyset$; /* new sequences need to add */
      foreach $S \in n.\mathcal{S}$ **do**
        if $L_r(R(x, Sc.id)) = 0$ **then** $\mathcal{S} = \mathcal{S} \cup \{Sc.id\}$; **else**
          $c.\mathcal{S} = c.\mathcal{S} \cup \{S\}$;
          /* 1. Removing zeros appearing after the last 1 */
          $S' = S-$ zeros appearing after the last 1;
          $\mathcal{S} = \mathcal{S} \cup \{S'c.id\}$;
          /* 2. Deleting ones in the sequence */
          $S'' = S - \{y \mid y \in S, R(x, y) = 1\}$;
          $\mathcal{S} = \mathcal{S} \cup \{S''c.id\}$;
      merger sequences in $\mathcal{S}$ with sequences already in node c;
      $c.visited = c.visited + 1$;
      if $c.NParent = c.visited$ **then** add c to the end of Q;
return $\mathcal{Y}$;

For the function of merging candidate sequences, there are three inputs: an element x of A, the current node c, and the collection of sequences $\mathcal{S}$. If there are existing sequences in the current node c, then the function will check if it is possible to combine the new sequences with existing sequences. If a new sequence is a subset of another existing sequence, then there is no further action required. If an existing sequence is a subset of a new sequence, then the old sequence will be removed. Because the sequences computed in this step are one part of the candidate closed bicluster, they must be maximal. When combining two sequences, the last common node is computed first. If the last common node is the last node of both sequences, then the second last common node is computed. Combination will be computed between two subsequences starting from the second last common node to the last common node of both original sequences. If the last common node is not the last node of both sequences, the combination will happen between two subsequences starting from the

last common node of each. The previous part of the sequence will not be considered, because the combination of them has already been evaluated when processing the second last common node or the last common node. When the two subsequences are completed un-ordered, the two sequences can be merged only when the Lazarus count for any possible order is 0. When both sequences have some zeros and ones, there are multiple ways of combinations: combining only ones of both sequences; combining all zeros of one sequence with one 1 of the other sequence or combining all zeros of one sequence with one 1 of the other.

Algorithm 4.3 computes the closed biclusters by extending the A part of the AB_pairs computed in the previous steps. The way of extending is similar to the method used in Algorithm 4.2. Here the extension is conducted by following the Hasse diagram of the poset $(A, \leq_A)$. Handling the case when Lazarus events happen is more complicated. If combining the two AB_pairs causes a Lazarus event, there are three ways to make modifications: deleting columns (elements from the sequence of B); deleting rows (elements from the sequence of A); and deleting both some columns and some rows that cause Lazarus events. In the end, the algorithm will compute combinations of the new AB_pair with any existing AB_pairs of the current node. The rules of combining two AB_pairs are similar to combining two sequences as described above.

Algoritmo 4.3 $Extension$: Extending A parts of AB_pairs.

Input: Hasse diagram of A, H_A
Input: AB_pairs with single element in A part, $\mathcal{P}$
Output: Closed biclusters $\mathcal{C}$
$\mathcal{C} = \emptyset$; $Q = \emptyset$;
foreach *root node* n *of* H_A **do**
  $n.\mathcal{S} = n.\mathcal{S} \cup n.id \times \mathcal{P}(n.id)$;
  add n to the end of Q;
while $Q \neq \emptyset$ **do**
  $n \leftarrow$ the first entry of Q;
  if $n.children = \emptyset$ **then**
    $\mathcal{C} = \mathcal{C} \cup n.\mathcal{S}$;
  else
    foreach $c \in n.children$ **do**
      $\mathcal{Y}_c = \mathcal{P}(c.id)$;
      foreach $X \times \mathcal{Y} \in n.\mathcal{S}$ **do**
        foreach $Y \in \mathcal{Y}$ *and* $Y_c \in \mathcal{Y}_c$ **do**
          if $Y \cap Y_c = \emptyset$ **then**
            $\mathcal{S} = \mathcal{S} \cup X \times Y$;
          else
            $S = Y \cap Y_c$;
            `/* lazarus count is zero */`
            if $L_c(R(Xc.id, S)) = 0$ **then** $\mathcal{S} = \mathcal{S} \cup X \times Y$; **else**
              $\mathcal{S} = \mathcal{S} \cup X \times Y$;
              generate different combinations of rows or columns $X' \times Y'$;
              $\mathcal{S} = \mathcal{S} \cup X' \times Y'$;
      merge candidate AB_pairs in $\mathcal{S}$ with those already in node c;
      $c.visited = c.visited + 1$;
      if $c.NParent = c.visited$ **then** add c to the end of Q
foreach $C \in \mathcal{C}$ **do** remove rows and columns with all zeros;
return $\mathcal{C}$;

4.5 Experimental Results

To illustrate the applicability of our approach, we have applied our algorithm to study the coupling of co-regulated genes and metabolic reactions in *E.coli*. The *E.coli K-12* transcriptional regulation data set obtained from publicly accessible database RegulonDB [24] was used. This dataset also contains the sequence position of each gene in the genome. The metabolic pathways are collected from the Kyoto Encyclopedia of Genes and Genomes database (KEGG) [48]. The characteristics of these datasets are listed in Table 4.1.

The partial order of reactions is derived by following the directions of reactions in the metabolic

Table 4.1: Characteristics of datasets used in this work.

Type	Operons/Pathway	Genes/Enzymes	Reactions/Regulations
Transcriptional regulation	2,665	4,576	3,013
Metabolic pathway	85	1,347	1,023

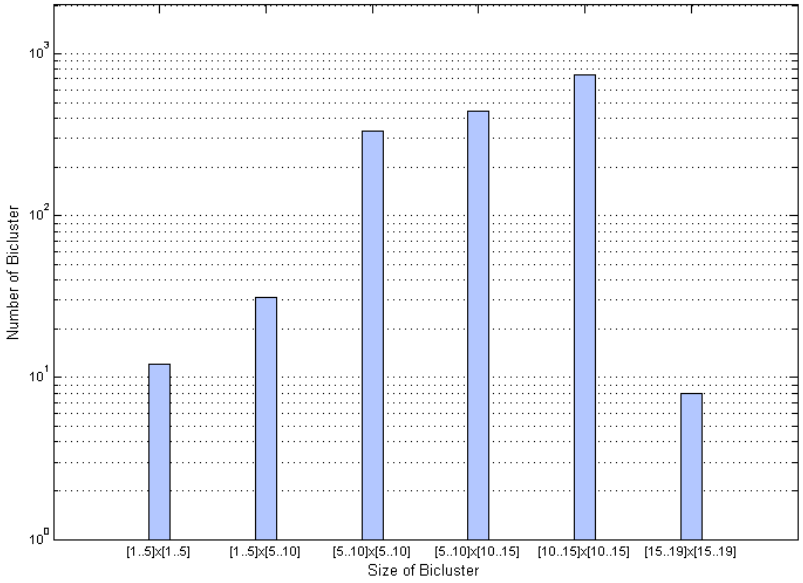


Figure 4.10: Number of biclusters of each size. These biclusters are functionally enriched with p-value 0.001.

pathway. Thus Reaction A precedes Reaction B if the products of A are the reactants of B. Operons are ordered according to the physical locations of genes in the genome. The order of operons also forms a partial order, because some of the operons share common genes. Reactions and operons form a binary relation, i.e., if the genes in an operon code for enzymes that catalyze a reaction, then the operon and the reaction has a relation, otherwise not. We computed biclusters over this reaction-operon matrix. Figure 4.10 depicts the number of new biclusters for each size range. The biclusters listed in this figure are statistically significant with respect to the functional annotations of genes in the bicluster. Because of the structure of Gene Ontology [3], when computing the over-represented GO terms of a given gene set, the parent-child relationships between GO terms need to be considered. Otherwise, more specific terms under known over-represented terms tend to be falsely detected as over-represented as well. We used the strategy proposed by Grossman et al. [32] that takes the parent-child relationships into account. Figure 4.11 shows the cumulative density distribution of the biclusters. For example, the y-axis value when $x = 0.4$ indicates the number of biclusters whose densities are equal to or less than 0.4. The densities of most biclusters range from 0.07 to 0.2.

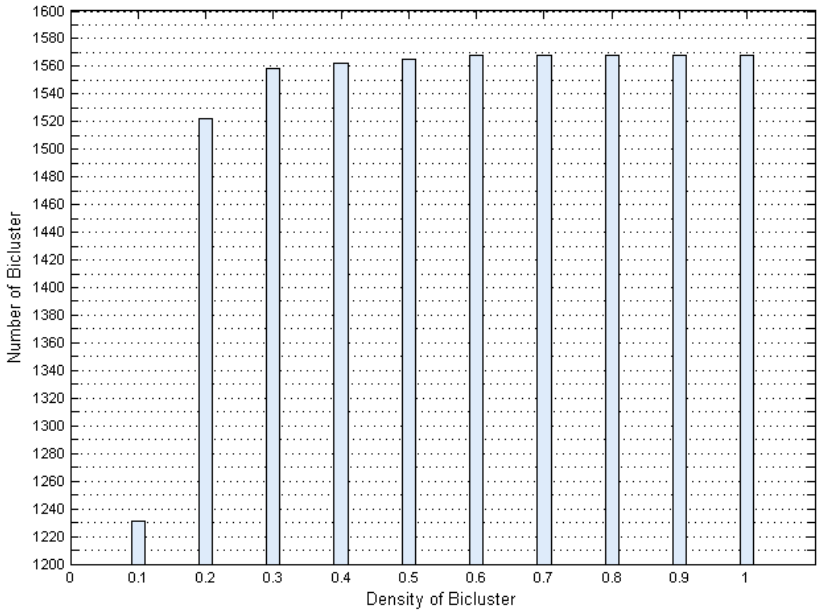


Figure 4.11: Density of biclusters that are functionally enriched with p-value 0.001.

Figure 4.12 shows three biclusters computed from the above dataset. Genes are color-coded according to which operons they belong to. The solid arrow in the figure means that the two genes/reactions are consecutive. The dotted arrow means that there are other genes/reactions in between. The solid lines between genes and reactions represent genes coding for enzymes that catalyze the corresponding reactions. The first two biclusters show examples where the physical locations of genes on the DNA sequences match to the order of metabolic reactions that are catalyzed by their products. In the first bicluster, genes belonging to the same operon code for enzymes catalyzing consecutive reactions. In the second bicluster, operons located upstream in the DNA sequences code for enzymes catalyzing reactions also upstream in the metabolic reactions. The third bicluster shows another case where the locations of genes in the genome do not match the order of reactions catalyzed by their products. In this third bicluster, upstream genes code for enzymes that catalyze the reactions in the middle, but the subsequent genes are connecting with upstream reactions.

We also calculated the number of biclusters exclusively in a single pathway. The results are listed in Table 4.2. The remaining biclusters contain reactions of more than one pathways. Purine metabolism contains most of the exclusive biclusters that appear in one pathway. This is consistent with the fact that purine metabolism contains a large number of reactions.

We computed the reaction coupling of co-regulated genes, i.e., co-regulated genes appearing in the same biclusters, and compared these with the results of a previous study on the flux coupling of co-regulated genes [63]. In this study, the flux coupling of co-regulated genes are computed

Table 4.2: Number of biclusters in the same pathway.

Pathway	Number of bicluster
GlycolysisGluconeogenesis	7
Fructose and mannose metabolism	2
Purine metabolism	278
Peptidoglycan biosynthesis	3
Butanoate metabolism	3

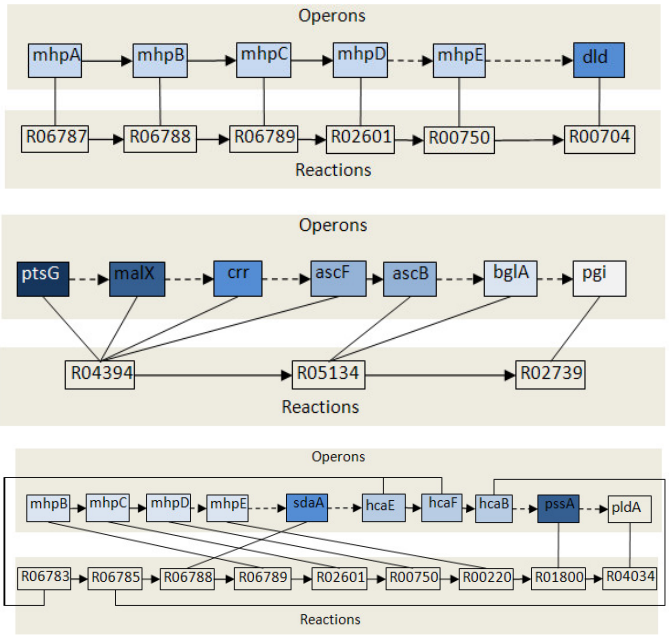


Figure 4.12: Examples of biclusters. Solid line: direct connection, Dashed line: indirect connection.

Table 4.3: Coupling of co-regulated genes.

Study	Common pairs	Total pairs
Biclusters with partial order information	64	1530
Flux coupling	64	191

based on flux balance analysis techniques. Table 4.3 shows the comparison results. Our approach was able to re-produce 30% of the flux coupling co-regulated gene pairs that have been reported in the previous study. Many other potentially interesting reactions coupling gene pairs were also discovered.

4.6 Discussion

We have shown how a more expressive notion of biclusters leads to biologically relevant results for understanding the interplay between operon structure and reaction pathways. Further work will focus on domain enrichment of the biclusters and using the proposed algorithms for genome-genome comparisons.

Chapter 5

Conclusion and Future Work

This dissertation has presented new algorithms for mining biological network datasets. These algorithms have been motivated by current topics in bioinformatics research: discovering unknown gene-phenotype relationships and understanding of the coupling of co-regulated genes in metabolic networks. Empirical results on real datasets have shown the applicability and effectiveness of the proposed methodologies.

5.1 Contributions

The specific contributions of our work can be summarized as follows.

Relative Importance Methods.

- **Prior state-of-the-art:** Relative importance methods have been originally proposed by White and Smyth [86].
- **Our contributions:** We have adapted the originally published relative importance methods to mining problems where two types of networks are supplied: an undirected network and an undirected bipartite network. In the original work of White and Smyth, only a single (undirected) network was used. Our proposed algorithm has also been applied to predict unknown relationships between genes and phenotypes. Relationships between phenotypes are derived first, and a prediction step is defined based on the relations between phenotypes just inferred. Experiments on *C.elegans* early embryogenesis and *S. cerevisiae* deletion mutants datasets show that our algorithm consistently outperforms other approaches. The strong results indicate that the combination of gene networks and phenotype profiles provides a powerful synergy that is not obtainable with either method alone. To explain the success of

our framework, we have further discussed several aspects of the algorithm. We have theoretically proved the convergence of the algorithm. The sensitivity of the initial rank has been discussed in detail. The problem of finding the optimal initial rank (i.e., the initial rank can produce the best prediction accuracy) has been formulated as a semidefinite programming problem. Further, we have studied the improvement afforded by the relative importance method as compared to a global importance formulation.

Constrained Graph Minimal Separators.

- **Prior state-of-the-art:** The problem of computing minimal separators of a graph has been studied for many years [7, 52]. Berry and Sigyret [8] have built an one-to-one mapping between minimal separators of a co-bipartite graph and the elements of the corresponding concept lattice.
- **Our contributions:** We have generalized the traditional problem of finding minimal separators to the problem of mining constrained minimal separators, where some parts of the graph are intended to be disconnected from each other whereas others are expected to be as connected as possible. We have loosen the requirement for a co-bipartite graph to a more general bipartite graph. We have constructed the mapping between a concept and a constrained minimal separator of a general bipartite graph and theoretically prove it. We have presented an efficient algorithm to compute constrained minimal separators and proved its correctness. This algorithm transforms the given problem instance so that the computation is conducted over a smaller graph than the original network of interactions. This approach has been applied to model gene perturbation screens by viewing the perturbed genes as a *graph separator* in *S. cerevisiae* and *C. elegans*. The sub-networks that our case studies separate and the phenotypes that result have been validated by published research. Ours is the first work that provides a computational approach that can be used to model multiple gene perturbations.

Mining Biclusters with Supplied Partial Orders.

- **Prior state-of-the-art:** No formulation exists that infers biclusters over domains with partial orders.
- **Our contributions:** We have generalized the notion of biclusters over sets to biclusters over partially ordered sets. Our work is the first to consider the order information of entities in forming biclusters. We have formally defined a closure operator in this new space and proved its correctness. The new definition includes the case where the order information is not given, thus subsuming the classical definition of biclusters. An efficient algorithm has been designed to compute the new biclusters. Experiments on *E.coli* metabolic networks and transcriptional regulations datasets have discovered many biologically interesting patterns of coupling between co-regulated genes and metabolic reactions.

5.2 Future Work

Our work can be extended in many algorithmic directions as well as to many new applications.

The success of the relative importance method and the close-to-optimal setting of the initial weights deserves further study. This approach can be investigated for other problem domains such as recommender systems and social networks which rely on the situation of a node in a graph to make recommendations or predictions.

The constrained graph minimal separator method can be extended to consider many different types of constraints. Posing separator problems over multiple graphs simultaneously is an interesting direction. Another direction is to develop methods to mine constraint minimal separators over directed graphs such as metabolic networks. The measure of the influence of the discovered minimal separator to the 'other' parts of the network can be further studied. In terms of new applications, this work finds uses in domains such as social networks and pandemic disease modeling (e.g., quarantining a subset of the population to prevent infections from spreading).

Finally, the new notion of biclusters developed here can be used in a compositional data mining framework [43] where we can compose together biclusters from multiple relations together to form long chains of inference. This work will find applications in many biological applications involving richer forms of datasets.

Bibliography

[1] F. Alizadeh, J.-P. A. Haeberly, M. V. Nayakkankuppam, M. L. Overton, and S. S., "SDPpack users guide ł version 0.9 beta for Matlab 5.0," New York University, Tech. Rep. TR1997-737, June 1997.

[2] S. Annunen, J. Körkkö, M. Czarny, M. Warman, H. Brunner, H. Kääriäinen, J. Mulliken, L. Tranebjaerg, D. Brooks, G. Cox, J. Cruysberg, M. Curtis, S. Davenport, C. Friedrich, I. Kaitila, M. Krawczynski, A. Latos-Bielenska, S. Mukai, B. Olsen, N. Shinno, M. Somer, M. Vikkula, J. Zlotogora, D. Prockop, and L. Ala-Kokko, "Splicing mutations of 54-bp exons in the *COL11A1* gene cause Marshall syndrome, but other mutations cause overlapping Marshall/Stickler phenotypes," *Am J Hum Genet*, vol. 65, no. 4, pp. 974–983, October 1999.

[3] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, M. Harris, D. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. Matese, J. Richardson, M. Ringwald, G. Rubin, and G. Sherlock, "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium," *Nat Genet*, vol. 25, no. 1, pp. 25–29, May 2000.

[4] A. Bairoch, "The ENZYME database in 2000," *Nucl. Acids Res.*, vol. 28, no. 1, pp. 304–305, January 2000.

[5] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *J Comput Biol*, vol. 6, no. 3/4, pp. 281–297, 1999.

[6] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Matrmatical Sciences*. Academic Press, 1979.

[7] A. Berry, J. Bordat, and O. Cogis, "Generating All the Minimal Separators of a Graph," *IJFCS*, vol. 11, no. 3, pp. 397–403, 2000.

[8] A. Berry and A. Sigayret, "Representing a concept lattice by a graph," in *Proceedings of the 2nd SIAM Workshop on Data Mining, Workshop on Discrete Mathematics for Data Mining*, 2002.

[9] A. Berry, "Treillis de Galois des Separateurs Minimaux d'un graphe non-orient," in *Master's Dissertation, LIRMM*, 1995.

[10] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Inter. Tech.*, vol. 5, no. 1, pp. 92–128, February 2005.

[11] D. Botstein and N. Risch, "Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease," *Nat Genet*, vol. 33 Suppl, pp. 228–237, March 2003.

[12] B. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, K. Dolinski, and M. Tyers, "The BioGRID Interaction Database: 2008 update," *Nucleic Acids Res*, vol. 36, January 2008.

[13] T. Brummelkamp and R. Bernards, "New tools for functional mammalian cancer genetics," *Nat Rev Cancer*, vol. 3, no. 10, pp. 781–789, October 2003.

[14] V. Cherkasova, D. Lyons, and E. Elion, "Fus3p and Kss1p Control G1 Arrest in Saccharomyces cerevisiae Through a Balance of Distinct Arrest and Proliferative Functions That Operate in Parallel With Far1p," *Genetics*, vol. 151, no. 3, pp. 989–1004, 1999.

[15] J. Cherry, C. Adler, C. Ball, S. Chervitz, S. Dwight, E. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, "SGD: Saccharomyces Genome Database," *Nucleic Acids Res*, vol. 26, no. 1, pp. 73–79, January 1998.

[16] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. Cambridge University Press, April 1990.

[17] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2001, pp. 269–274.

[18] C. Ding, H. Zha, X. He, P. Husbands, and H. Simon, "Link Analysis: Hubs and Authorities on the World Wide Web," *SIAM Review*, vol. 46, pp. 256–268, June 2004.

[19] G. Dirac, "On rigid circuit graphs," *Abh Math Sem*, vol. 25, pp. 71–76, 1961.

[20] A. Farahat, T. Lofaro, J. C. Miller, G. Rae, and L. A. Ward, "Authority Rankings from HITS, PageRank, and SALSA: Existence, Uniqueness, and Effect of Initialization," *SIAM J. Sci. Comput.*, vol. 27, no. 4, pp. 1181–1201, 2006.

[21] A. Fire, S. Xu, M. Montgomery, S. Kostas, S. Driver, and C. Mello, "Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans," *Nature*, vol. 391, no. 6669, pp. 806–811, 1998.

[22] H. Fraser and J. Plotkin, "Using protein complexes to predict phenotypic effects of gene mutation," *Genome Biol*, vol. 8, pp. R252+, 2007.

[23] N. Freimer and C. Sabatti, "The Human Phenome Project," *Nat Genet*, vol. 34, no. 1, pp. 15–21, May 2003.

[24] S. Gama-Castro, V. Jiménez-Jacinto, M. Peralta-Gil, A. Santos-Zavaleta, M. I. Peñaloza Spinola, B. Contreras-Moreira, J. Segura-Salazar, L. Muñiz Rascado, I. Martínez-Flores, H. Salgado, C. Bonavides-Martínez, C. Abreu-Goodger, C. Rodríguez-Penagos, J. Miranda-Ríos, E. Morett, E. Merino, A. M. Huerta, L. Treviño Quintanilla, and J. Collado-Vides, "RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation." *Nucleic acids research*, vol. 36, no. Database issue, January 2008.

[25] B. Ganter and R. Wille, *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999.

[26] F. R. Gantmacher, *Matrix Theory*. Chelsea Publishing Company, 1974, vol. 2.

[27] S. Geisser, "The predictive sample reuse method with applications," *Journal of the American Statistical Association*, vol. 70, no. 350, pp. 320–328, 1975.

[28] G. Giaever, A. Chu, L. Ni, C. Connelly, L. Riles, S. Véronneau, S. Dow, A. Lucau-Danila, K. Anderson, B. André, A. Arkin, A. Astromoff, M. El-Bakkoury, R. Bangham, R. Benito, S. Brachat, S. Campanaro, M. Curtiss, K. Davis, A. Deutschbauer, K. Entian, P. Flaherty, F. Foury, D. Garfinkel, M. Gerstein, D. Gotte, U. Güldener, J. Hegemann, S. Hempel, Z. Herman, D. Jaramillo, D. Kelly, S. Kelly, P. Kötter, D. LaBonte, D. Lamb, N. Lan, H. Liang, H. Liao, L. Liu, C. Luo, M. Lussier, R. Mao, P. Menard, S. Ooi, J. Revuelta, C. Roberts, M. Rose, P. Ross-Macdonald, B. Scherens, G. Schimmack, B. Shafer, D. Shoemaker, S. Sookhai-Mahadeo, R. Storms, J. Strathern, G. Valle, M. Voet, G. Volckaert, C. Wang, T. Ward, J. Wilhelmy, E. Winzeler, Y. Yang, G. Yen, E. Youngman, K. Yu, H. Bussey, J. Boeke, M. Snyder, P. Philippsen, R. Davis, and M. Johnston, "Functional profiling of the Saccharomyces cerevisiae genome," *Nature*, vol. 418, no. 6896, pp. 387–391, 2002.

[29] M. Golumbic, "Algorithmic graph theory and perfect graphs," *Academic Press*, 1980.

[30] A. Graham, *Nonnegative Matrices and Applicable Topics in Linear Algebra*. Ellis Horwood Limited, 1987.

[31] A. Griffith, L. Sprunger, D. Sirko-Osadsa, G. Tiller, M. Meisler, and M. Warman, "Marshall syndrome associated with a splicing defect at the *COL11A1* locus," *Am J Hum Genet*, vol. 62, no. 4, pp. 816–823, April 1998.

[32] S. Grossmann, S. Bauer, P. Robinson, and M. Vingron, "An improved statistic for detecting over-represented Gene Ontology annotations in gene sets," *Proceedings of the Lecture Notes in Computer Science*, vol. 3909, pp. 85–98, 2006.

[33] G. Grothaus, A. Mufti, and T. Murali, "Automatic Layout and Visualization of Biclusters," *Algorithms Mol Biol*, vol. 1, no. 15, 2006.

[34] U. Gueldener, M. Muensterkoetter, G. Kastenmueller, N. Strack, J. van Helden, C. Lemer, J. Richelles, S. J. Wodak, G. J. Martinez, P. J. Ortin, H. Michael, A. Kaps, E. Talla, B. Dujon, B. Andre, J. Souciet, J. De Montigny, E. Bon, C. Gaillardin, and H. Mewes, "CYGD: the Comprehensive Yeast Genome Database," *Nucleic Acids Res*, vol. 33, January 2005.

[35] K. Gunsalus, H. Ge, A. Schetter, D. Goldberg, J. Han, T. Hao, G. Berriz, N. Bertin, J. Huang, L. Chuang, N. Li, R. Mani, A. Hyman, B. Sönnichsen, C. Echeverri, F. Roth, M. Vidal, and F. Piano, "Predictive models of molecular machines involved in Caenorhabditis elegans early embryogenesis," *Nature*, vol. 436, no. 7052, pp. 861–865, 2005.

[36] G. J. Hannon, "RNA interference," *Nature*, vol. 418, no. 6894, pp. 244–251, July 2002.

[37] T. H. Haveliwala, "Efficient computation of PageRank," Stanford Digital Library Technologies Project, Tech. Rep. 1999-31, 1999.

[38] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 1999, pp. 230–237.

[39] Herlocker, J. and Konstan, J. and Borchers, A. and Riedl, J. , "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '99)*. ACM Press, 1999, pp. 230–237.

[40] P. G. Hoel, S. C. Port, and C. J. Stone, *Introduction to stochastic processes*. Houghton Mi2in Company, 1972.

[41] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. Wiley, 1973.

[42] J. Jacob, M. Jentsch, D. Kostka, S. Bentink, and R. Spang, "Detecting hierarchical structure in molecular characteristics of disease using transitive approximations of directed graphs." *Bioinformatics*, February 2008.

[43] Y. Jin, T. M. Murali, and N. Ramakrishnan, "Compositional mining of multirelational biological datasets," *ACM Trans Knowl Discov Data*, vol. 2, no. 1, pp. 1–35, 2008.

[44] Y. Jin, N. Ramakrishnan, L. S. Heath, and R. F. Helm, "Using relative importance methods to model high-throughput gene perturbation screens," *the 7th Annual International Conference on Computational Systems Bioinformatics (CSB'08)*, 2008.

[45] Y. Jin, N. Ramakrishnan, L. S. Heath, and R. F. Helm, "Constrained Mining of Minimal Separators with Applications to Gene Perturbation Studies," in *Proceedings of the 7th Asia Pacific Bioinformatics Conference (APBC'09)*, January 2009.

[46] S. D. Kamvar, T. H. Haveliwala, and G. H. Golub, "Adaptive methods for the computation of PageRank," Stanford University, Tech. Rep. 2003-26, 2003.

[47] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Exploiting the block structure of the Web for computing PageRank," Stanford University, Tech. Rep. 2003-17, 2003.

[48] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes," *Nucleic Acids Res*, vol. 28, no. 1, pp. 27–30, 2000.

[49] P. D. Karp, C. A. Ouzounis, C. Moore-Kochlacs, L. Goldovsky, P. Kaipa, D. Ahrén, S. Tsoka, N. Darzentas, V. Kunin, and N. López-Bigas, "Expansion of the BioCyc collection of pathway/genome databases to 160 genomes." *Nucleic acids research*, vol. 33, no. 19, pp. 6083–6089, 2005.

[50] M. G. Kendall, "A New Measure of Rank Correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, June 1938.

[51] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *JACM*, vol. 46, no. 5, pp. 604–632, 1999.

[52] T. Kloks and D. Kratsch, "Finding all minimal separators of a graph," *STACS*, pp. 759–768, 1994.

[53] K. Lage, O. Karlberg, Z. Storling, P. I. Ólason, A. Pedersen, O. Rigina, A. Hinsby, Z. Tümer, F. Pociot, N. Tommerup, Y. Moreau, and S. Brunak, "A human phenome-interactome network of protein complexes implicated in genetic disorders," *Nat Biotechnol*, vol. 25, no. 3, pp. 309–316, 2007.

[54] G. F. Lawler, *Introduction to stochastic processes*. CRC Press, 2006.

[55] C. P.-C. Lee, G. H. Golub, and S. A. Zenios, "A fast two-stage algorithm for computing PageRank and its extensions," Scientific Computation and Computational Mathematics, Stanford University, Tech. Rep. SCCM-2003-15, 2003.

[56] I. Lee, B. Lehner, C. Crombie, W. Wong, A. Fraser, and E. Marcotte, "A single gene network accurately predicts phenotypic effects of gene perturbation in Caenorhabditis elegans," *Nat Genet*, vol. 40, no. 2, pp. 181–188, 2008.

[57] X. Ma, H. Lee, and F. Sun, "CGI: a new approach for prioritizing genes by combining gene expression and proteinprotein interaction data," *Bioinformatics*, vol. 23, no. 2, pp. 215–221, 2007.

[58] S. Madeira and A. Oliveira, "Biclustering algorithms for biological data analysis: a survey," *IEEE/ACM Trans Comput Biol Bioinf*, vol. 1, no. 1, pp. 24–45, January 2004.

[59] F. Markowetz, D. Kostka, O. Troyanskaya, and R. Spang, "Nested effects models for high-dimensional phenotyping screens," *Bioinformatics*, vol. 23, no. 13, pp. 305–312, 2007.

[60] H. W. Mewes, D. Frishman, K. Mayer, M. Münsterkötter, O. Noubibou, P. Pagel, T. Rattei, M. Oesterheld, A. Ruepp, and V. Stümpflen, "MIPS: analysis and annotation of proteins from whole genomes in 2005," *Nucleic Acids Res*, vol. 34, 2006.

[61] J. Morrison, R. Breitling, D. Higham, and D. Gilbert, "GeneRank: Using search engine technology for the analysis of microarray experiments," *BMC Bioinformatics*, vol. 6, no. 1, 2005.

[62] I. Muñoz, E. Simón, N. Casals, J. Clotet, and J. Ariño, "Identification of multicopy suppressors of cell cycle arrest at the G1-S transition in Saccharomyces cerevisiae," *Yeast*, vol. 20, no. 2, pp. 157–169, 2003.

[63] R. A. Notebaart, B. Teusink, R. J. Siezen, and B. Papp, "Co-regulation of metabolic genes is better explained by flux coupling than by network distance," *PLoS Computational Biology*, vol. 4, no. 1, pp. e26+, January 2008.

[64] Y. Ohya, J. Sese, M. Yukawa, F. Sano, Y. Nakatani, T. Saito, A. Saka, T. Fukuda, S. Ishihara, S. Oka, G. Suzuki, M. Watanabe, A. Hirata, M. Ohtani, H. Sawai, N. Fraysse, J. Latge, J. Francois, M. Aebi, S. Tanaka, S. Muramatsu, H. Araki, K. Sonoike, S. Nogami, and S. Morishita, "High-dimensional and large-scale phenotyping of yeast mutants," *PNAS*, vol. 102, no. 52, pp. 19 015–19 020, December 2005.

[65] M. Oti and H. Brunner, "The modular nature of genetic diseases," *Clin Genet*, vol. 71, no. 1, pp. 1–11, January 2007.

[66] M. Oti, M. Huynen, and H. Brunner, "Phenome connections," *Trends Genet*, vol. 24, no. 3, pp. 103–106, March 2008.

[67] M. Oti, B. Snel, M. A. Huynen, and H. G. Brunner, "Predicting disease genes using protein-protein interactions." *J Med Genet*, vol. 43, no. 8, pp. 691–698, 2006.

[68] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998.

[69] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules," *Lecture Notes in Computer Science*, vol. 1540, pp. 398–416, 1999.

[70] A. Pati, Y. Jin, K. Klage, R. Helm, L. Heath, and N. Ramakrishnan, "CMGSDB: integrating heterogeneous Caenorhabditis elegans data sources using compositional data mining," *Nucleic Acids Res*, vol. 36, 2008.

[71] J. Pei, J. Han, and R. Mao, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets," in *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000, pp. 21–30.

[72] F. Piano, A. Schetter, D. Morton, K. Gunsalus, V. Reinke, S. Kim, and K. Kemphues, "Gene clustering based on RNAi phenotypes of ovary-enriched genes in *C. elegans*," *Curr Biol*, vol. 12, no. 22, pp. 1959–1964, November 2002.

[73] A. Rosenberg and L. Heath, "Graph separators, with applications," *Kluwer Academic/Plenum Publishers*, 2001.

[74] D. Ruths, L. Nakhleh, M. Iyengar, A. Reddy, and P. Ram, "Hypothesis generation in signaling networks," *J Comput Biol*, vol. 13, no. 9, pp. 1546–1557, 2006.

[75] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *World Wide Web*, 2001, pp. 285–295.

[76] B. Scherens and A. Goffeau, "The uses of genome-wide yeast mutant collections," *Genome Biol*, vol. 5, no. 7, p. 229, 2004.

[77] S. Siegel and N. J. Castellan, *Nonparametric statistics for the behavioral sciences*. McGraw-Hill Book Company, 1988.

[78] V. Smith, K. Chou, D. Lashkari, D. Botstein, and P. Brown, "Functional analysis of the genes of yeast chromosome V by genetic footprinting," *Science*, vol. 274, no. 5295, pp. 2069–2074, 1996.

[79] M. Snead and J. Yates, "Clinical and molecular genetics of Stickler syndrome," *J Med Genet*, vol. 36, no. 5, pp. 353–359, May 1999.

[80] B. Sönnichsen, L. Koski, A. Walsh, P. Marschall, B. Neumann, M. Brehm, A. Alleaume, J. Artelt, P. Bettencourt, E. Cassin, M. Hewitson, C. Holz, M. Khan, S. Lazik, C. Martin, B. Nitzsche, M. Ruer, J. Stamford, M. Winzi, R. Heinkel, M. Röder, J. Finell, H. H., S. Jones, M. Jones, F. Piano, K. Gunsalus, K. Oegema, P. Gönczy, A. Coulson, A. Hyman, and C. Echeverri, "Full-genome RNAi profiling of early embryogenesis in Caenorhabditis elegans," *Nature*, vol. 434, no. 7032, pp. 462–469, 2005.

[81] M. Stone, "Cross-validation and multinomial prediction," *Biometrika*, vol. 61, no. 3, pp. 509–515, December 1974.

[82] A. Ukkonen, M. Fortelius, and H. Mannila, "Finding partial orders from unordered 0-1 data," in *Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining(KDD '05)*. New York, NY, USA: ACM, 2005, pp. 285–293.

[83] N. Valtz, M. Peter, and I. Herskowitz, "FAR1 is required for oriented polarization of yeast cells in response to mating pheromones," *J. Cell Biol*, vol. 131, no. 4, pp. 863–873, 1995.

[84] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, 1996.

[85] J. T. L. Wang, M. J. Zaki, and H. T. T. Toivonen, *Data Mining in Bioinformatics (Advanced Information and Knowledge Processing)*. Springer, September 2004.

[86] S. White and P. Smyth, "Algorithms for estimating relative importance in networks," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'03)*. ACM Press, 2003, pp. 266–275.

[87] H. Yu and M. Gerstein, "Genomic analysis of the hierarchical structure of regulatory networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 40, pp. 14 724–14 731, October 2006.

[88] M. Zaki and C. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining," in *Proceedings of the 2nd SIAM International Conference on Data Mining(SIAM'02)*, 2002.

[89] M. J. Zaki, "Generating non-redundant association rules," in *Knowledge Discovery and Data Mining*, 2000, pp. 34–43.