University of Alberta

BIO-RELATION DISCOVERY AND SPARSE LEARNING

by

Yi Shi

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Fall 2012
Edmonton, Alberta

Abstract

In this dissertation, I discuss several important problems in the area of bio-relation discovery (BRD). Discovering bio-relations is an important problem that arises frequently in bioinformatics. It involves identifying relationships (usually pairwise) between bio-entities. These relationships can be categorized as undirected versus directed. I will investigate both types of BRD problems in this dissertation. For undirected BRD, I will specifically discuss the gene-sample expression biclustering problem. For directed BRD, I will focus on problems in gene regulatory network inference and drug-target network inference. For all these problems I have investigated both heuristic approaches and sparse learning based approaches from machine learning.

Acknowledgements

This thesis concludes the work and research in my Ph.D. life for the past five years. I have been very fortunate to have the support and encouragement from many people during this important period of my life. Here I would like to take this opportunity to express my sincere appreciation to each of them.

I am deeply grateful to my supervisors, Dr. Dale Schuurmans and Dr. Guohui Lin. Throughout my research, Dr. Schuurmans has been offering tremendous and insightful guidance and help. I have benefited significantly from his broad knowledge and active research attitude, and I really appreciate how supportive and considerate Dale has been throughout this process. Dr. Lin has been a supportive supervisor and friend since my Masters and he is always ready to help me in various aspects; he not only teaches me how to do research, but also teaches me how to be a person.

I would like to express my appreciation to my committee members, Dr. Liang Li, Dr. Randy Goebel, Dr. Russell Greiner for their time, effort, and precious suggestions that helped my research to a great extend.

I am also thankful to my collaborators, Dr. Xiaoping Liao, Dr. Xinhua Zhang, Dr. Yuhong Guo, Dr. Nan Wang, Maryam Hasan, and Dr. Zhipeng Cai for their help in my research.

Moreover, my appreciation goes to my friends and colleagues for their generous support and great companionship. I was not lonely during the journey thanks to them. Finally, I want to thank my parents and my wife for their absolute and constant support. No matter how far away I am from them, they are always there for me. It is my great fortune in life to have them as my source of strength, happiness and love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the past few decades rapid developments in genomic and molecular analysis technologies have produced a tremendous amount of molecular biological information. Bioinformatics researchers therefore are facing the challenge of developing semi-automated analysis tools (e.g., pattern recognition, data mining, machine learning, and visualization tools) that can reliably and robustly interpret this information, to help improve our understanding of biological processes at a systems level. Major research topics in bioinformatics includes sequence alignment, gene finding, genome assembly, drug design, drug discovery, protein structure alignment, protein structure prediction, prediction of gene expression and protein-protein interactions, genome-wide association studies, and the modeling of evolution.

In this dissertation, I will investigate some important sub problems in these areas that involve bio-relation discovery (BRD). BRD is an important problem in bioinformatics where relationships between bio-entities need to be inferred [49]; such as gene-gene relation [17], DNA-protein interaction [63], protein-protein interaction [34], protein-compound interaction [93], etc. There are two categories of bio-relations that we are interested in inferring: undirected relations and directed relations. A relation is undirected if the two related entities are correlated to each other under a certain measurement, but we do not attempt to infer their causal relations. For example, genes that are regulated by a common transcription factor (TF) have undirected relations. A relation is directed if the two related entities have direct interaction or causal relation and the directedness is an important part of the analysis. For example, a transcription factor and the genes it regulates exhibit

directed relations.

In addressing these problems, I have investigated various heuristic approaches and approaches that are based on sparse learning methods. Sparse learning methods have recently attracted a lot of interest in areas of statistics, machine learning and signal processing. It turns out that many bio-relations discovery problems, either directed or undirected can also be represented as sparse learning problems. For example, a standard sparse learning model for regression can be formulated as:

$$\min_{\mathbf{w}} \sum_{i} L(y_i, \mathbf{w}^T \mathbf{x_i}) + \lambda \|\mathbf{w}\|_1 \tag{1.1}$$

where the function L is a loss function and $\lambda\|\mathbf{w}\|_1$ is a sparsity inducing regularizer, in this case the $L1$ norm regularizer. A well-known property of $L1$ norm regularization is that it encorages sparse solutions leading to loading vectors with many zeros, and thus performs model selection on top of regularization. Choosing different loss functions leads to different machine learning models. For example, if the least squares loss is chosen, (1.1) becomes a standard least absolute shrinkage and selection operator (LASSO) model [85]. If the hinge loss is chosen, (1.1) becomes a $L1$ norm support vector machine model [97]. In the work described below, I will ultimately formulate many of the BRD problems as sparse learning problems by exploiting sparsity inducing regularizers, and I will extend standard sparse learning algorithms to provide solutions to these problems, which we expect will improve on the quality of solutions produced by heuristic methods.

1.1 Undirected BRD

A very important undirected BRD problem in bioinformatics is to cluster genes based on their expression profiles, ensuring that genes of the same cluster are similar to each other and genes of different clusters are dissimilar to each other. Gene clustering can facilitate many downstream applications, including cluster-based gene selection and disease classification [9], and gene regulatory network inference [26]. A key question in gene clustering is how to define similarity between genes. Since two similar genes may present similar expression profiles only under a subset of observations (samples), one is often more interested in gene-sample bi-clustering

where both a subset of genes and a subset of samples have to be selected to identify a coherent clusters of activity. Bi-clustering, also known as co-clustering, or two-mode clustering is a more general problem than traditional clustering. It is a data mining technique that allows simultaneous clustering of the rows and columns of a matrix. The term was first invented by Mirkin [58] and introduced by Cheng and Church into gene expression microarray analysis [15]. In my work, I have investigated two heuristic bi-clustering algorithms, LCBD [72] (linear coherent bi-cluster discovery via line detection and sample majority voting) and an extension LinCoh [74] (linear coherent bi-cluster discovery via beam detection and sample set clustering), aiming to find genes that exhibit similar expression profiles over a subset of samples. I also investigated re-expressing bi-clustering as a sparse learning problem, which I will subsequently describe.

1.2 Directed BRD

One of the ultimate goals in system biology is to reconstruct the regulatory networks between genes. The regulatory relation discovery problem is an instance of directed BRD. It is well known that genes and their products cooperate with each other in the form of a dynamic gene regulatory network (GRN) that controls cell function. With the emergence of high-throughput gene profiling technology and ChIP-on-chip technology, GRN research has significantly advanced. A current goal of GRN research is to improve the fundamental understanding of how biological processes are coordinated in the cells. For this problem, I have developed a kernel-based learning model with structured sparse regularization to discover the causal control relationships between genes.

A second directed BRD problem I consider is drug-target prediction. In the drug-target prediction problem, we want to infer the binary relation between drugs and targets (proteins) based on known relations, drug structure information, and target sequence information. Chemical drugs are small molecules that bind to target proteins to change the protein conformation. Proteins usually operate as part of highly interconnected cellular networks referred to as interactome networks. When

their conformation are changed, their functions are subsequently changed and can result in treatment effects. Once an organism's gene regulatory network is in an abnormal state (e.g. via gene mutation, or environmental effect), disease can result. Drugs can help restore gene regulatory network into a normal state by regulating its targeted proteins. Therefore, the drug-target interaction discovery problem is another important directed BRD problem I will investigate in my research. In this dissertation, I will introduce a sparse learning method designed to address the drug-target network inference problem that is based on feature selection support vector machine classification which incorporates sparse learning and kernelization.

1.3 Overview

The remainder of this dissertation is organized as follows. Chapter 2 introduces the undirected BRD problem of bi-clustering in gene microarray data analysis. Here I present the two heuristic methods and one sparse learning based method. In Chapter 3, I investigate two directed BRD problems: the gene network inference problem and the drug-target interaction inference problem. A kernel-based regression model is proposed to address the gene regulatory network problem. A feature selection support vector machine approach that incorporates sparse learning regularizer and kernelization is investigated in the drug-target interaction inference problem.

Chapter 2

Undirected Bio-relation Discovery: Bi-clustering

DNA microarrays, also known as gene chips, DNA chips, or gene arrays, are a collection of microscopic DNA spots (probes) attached to a solid surface (e.g. glass, plastic or silicon chip). Expression profiling is a microarray technology that detects the RNAs that may or may not be translated into active proteins. They are used for the purpose of monitoring the expression levels of thousands of genes simultaneously over different conditions. In general, for a two-channel microarrays experiment, there are six steps for gene profiling. (1) Cells are extracted from different samples (for example, disease sample and normal sample) and cultivated in different tubes for a period of time so that adequate amount of different kinds of cells could be collected. (2) Messenger RNAs (mRNA) are isolated in different conditions and extracted from them. (3) mRNAs from different conditions are reverse transcribed into their corresponding cDNAs of different fluorescent dyes (cDNAs are artificially synthesized DNAs from mRNA templates). (4) Different cDNAs are mixed up together and a proportion of them are hybridized to a single microarray chip. (5) The hybridized microarray chip is read by scanners in different color channels (red and green) and their fluorescence intensities are collected and quantified by specific sensor and software. (6) The logarithmic ratios of the intensities from the two channels are computed and considered to be the gene expression levels. The one-channel microarray technique is similar to the two-channel technique, except that only one fluorescent is used and a reference condition is not required for an

experiment. Currently one-channel microarray techniques are more popular than two-channel techniques.

Although the microarray technology enables the language of biology to be spoken in mathematical terms, extracting useful information from the large volume of experimental microarray data remains a difficult challenge. One important problem in microarray analysis is to identify a subset of genes that have similar expression patterns under a common subset of conditions. Standard clustering methods, such as K-means clustering [31, 19], hierarchical clustering [79, 84] and self-organizing maps [82], are usually not suitable for microarray data analysis for two main reasons: (1) genes exhibit similar behaviors not over all conditions, but only over a subset of conditions, and (2) genes may participate in more than one functional processes and hence belong to multiple groups. Thus, traditional clustering algorithms typically do not produce a satisfactory solution. To overcome the limitations of the traditional clustering methods, the concept of bi-clustering was developed where one seeks groups of genes that exhibit similar expression patterns, but only over a subset of the sample conditions. Figure 2.1 illustrates a gene expression matrix without any obvious bi-clusters (left) and an expression matrix with a salient bi-cluster (right).

The term bi-clustering, also called co-clustering, or two-mode clustering was first mentioned by Hartigan in [30] and latter formalized by Mirkin in [58]. Cheng and Church [15] were the first to apply bi-clustering to gene expression analysis. Since then, dozens of bi-clustering algorithms have been proposed for the gene expression analysis. The general bi-clustering problem and many of its variants were proved to be NP-hard in [15], and therefore most bi-clustering algorithms comprise heuristic approaches unless special restrictions are made on the bi-cluster type and(or) bi-cluster structure. Among such bi-clustering algorithms, the majority assume that a expression matrix contains multiple bi-clusters rather than a single bi-cluster. Under the multiple bi-cluster circumstance, different bi-cluster structures can be considered, such as exclusive row and(or) column bi-clusters, checkerboard structure bi-clusters, non-overlapping tree-structured bi-clusters, non-overlapping non-exclusive bi-clusters, overlapping bi-clusters with hierarchical structure, arbi-

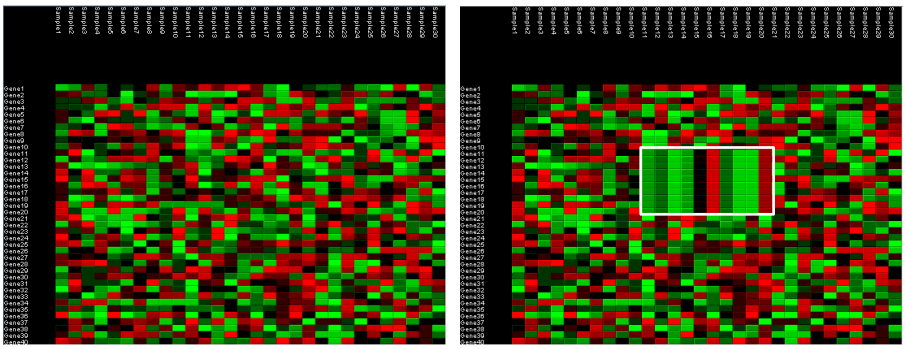


Figure 2.1: Example of a constant row bi-cluster in gene expression matrix. The left image shows a gene expression matrix without any obvious bi-clusters; the right image shows an expression matrix with a constant row bi-cluster.

trarily positioned overlapping bi-clusters, and arbitrarily positioned overlapping bi-clusters [55]. The specific form of bi-cluster that our algorithms are trying to detect is the last form of bi-cluster structure; i.e., arbitrarily positioned overlapping bi-clusters. This last form is a general structure that covers most of the other bi-cluster structures.

Before designing a bi-clustering algorithm, one needs to determine what type (model) of individual bi-clusters to be looking for. There are six primary types considered in the literature, illustrated in Figure 2.2: (a) the constant value model, (b) the constant row model, (c) the constant column model, (d) the additive coherent model, where each row or column is obtained by adding a constant to another row or column, (e) the multiplicative coherent model, where each row or column is obtained by multiplying another row or column by a constant value, and (f) the linear coherent model proposed by Gan et al. [22] where each row/column is obtained by multiplying another column by a constant value and then adding a constant. To understand which type of bi-cluster structure makes the sense for gene expression analysis, one should note that the ultimate purpose is to identify pairs of biologically related genes such that, under certain conditions [27, 2], one either activates or deactivates the other, or being commonly regulated by other genes during a genetic regulatory process. The goal of bi-clustering is to identify the undirected relations between genes. The result of the gene bi-clustering can facilitate many downstream

x	y	z	w
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0

(a)

x	y	z	w
1.2	1.2	1.2	1.2
0.8	0.8	0.8	0.8
1.5	1.5	1.5	1.5
0.6	0.6	0.6	0.6

(b)

x	y	z	w
1.2	0.8	1.5	0.6
1.2	0.8	1.5	0.6
1.2	0.8	1.5	0.6
1.2	0.8	1.5	0.6

(c)

x	y	z	w
1.2	0.8	1.5	0.6
1.0	0.6	1.3	0.4
2.0	1.6	2.3	1.4
0.7	0.3	1.2	0.3

(d)

x	y	z	w
2.0	4.0	8.0	1.0
1.0	2.0	4.0	0.5
4.0	8.0	16.0	2.0
1.0	2.0	4.0	0.5

(e)

x	y	z	w
2.0	4.0	3.0	5.0
1.5	2.5	2.0	3.0
2.3	4.3	3.3	5.3
4.5	8.5	6.5	10.5

(f)

Figure 2.2: Examples of different bi-cluster types: (a) constant value model; (b) constant row model; (c) constant column model; (d) additive coherent model; (e) multiplicative coherent model; (f) linear coherent model

applications such as clustering-based gene selection and disease classification [9] and gene regulatory network inference [26].

Because a gene may regulate (temporarily) a group of other genes, this problem becomes identifying groups of such genes, i.e., bi-clusters. Housekeeping genes, which are constitutively expressed over most conditions, are not biologically or clinically interesting. The genes that the first two bi-cluster models, i.e., (a) and (b) find tend to be this kind. Therefore, most existing algorithms are based on either the additive model (d) or the multiplicative model (e) [22]. Since type (f) is a more general type that unifies types (c), (d), and (e), I focus on seeking type (f) bi-clusters in this dissertation. Most bi-clustering algorithms implicitly address non time series microarray data and only few address time series microarray data [56]. For time series data, the time lag between mRNA transcription and transcription factor translation needs to be considered. In this dissertation, I focus on non time series data.

2.1 Approach 1: Line Detection

The first bi-clustering algorithm I investigated, the Linear Coherent Bi-cluster Discovering (LCBD) algorithm, is based on first detecting linear correlations between pairs of gene expression profiles, then identifying groups by sample majority voting. This work was published in [72].

2.1.1 Method

The LCBD algorithm is composed of three major steps. First, the gene pairwise linear relations are discovered based on the Hough transform [33], and the sample sets under which the gene pairs exhibit linear relations are recorded. Then, for each gene, a bi-cluster is constructed via sample majority voting based on the sample sets corresponding to that gene. Finally, the bi-clusters are refined locally and redundant bi-clusters are removed. A detailed description of the LCBD algorithm is given in Appendix A.

To evaluate the LCBD algorithm, I compared its performance to six existing, well known bi-clustering algorithms: Cheng and Church's algorithm, CC [15]; Samba [83]; Order Preserving Sub-matrix Algorithm, OPSM [3]; Iterative Signature Algorithm, ISA [36, 35]; Bimax [66]; and Maximum Similarity Bi-clusters of Gene Expression Data, MSBE [54]. The first five algorithms were selected and implemented in the survey [66]. The last algorithm, MSBE, is the first polynomial time bi-clustering algorithm that finds optimal solutions, but under certain constraints. To briefly explain each of the first five bi-clustering algorithms: in [15] Cheng and Church defined a merit score, called mean squared residue, to evaluate the quality of a bi-clustering, and then develop a greedy algorithm for finding δ-bi-clusters. Yang *et al*. improved Cheng and Church's method by allowing missing values in gene expression matrices. Tanay *et al*. [83] and Prelić *et al*. [66] search for bi-clusters of up-regulated or down-regulated expression values, while the original expression matrices are discretized to binary matrices during a pre-processing phase. Ihmels *et al*. [36, 35] used gene and condition signatures to evaluate bi-clusters, and propose a random iterative signature algorithm (ISA) when no prior

information of the matrix is available. Ben-Dor *et al.*[3] attempt to find the order-preserving sub-matrix (OPSM) bi-clusters in which all genes have same linear ordering, based on a heuristic algorithm.

I tested the algorithm on both synthetic datasets and two real datasets *Saccharomyces Cerevisiae* and *Arabidopsis Thaliana*. For the synthetic datasets, I evaluate the algorithms based on how well they identify the real bi-clusters embedded in the expression matrix beforehand. The size of the synthetic matrices and the size of the ground truth bi-clusters are chosen in consideration of (a) ensuring the gene/sample ratio is similar to real data cases, and (b) avoiding overly long run times (since the synthetic experiments are repeated many times).

I adopt the Prelić's match score function [66] as a quantified evaluation of merit: Let C, C^* be two sets of bi-clusters. The gene match score of C with respect to C^* is given by the function

$$\text{score}_G(\mathcal{C}, \mathcal{C}^*) = \frac{1}{|\mathcal{C}|} \sum_{(G_1,S_1)\in\mathcal{C}} \max_{(G_1^*,S_1^*)\in\mathcal{C}^*} \frac{|G_1 \cap G_1^*|}{|G_1 \cup G_1^*|} \tag{2.1}$$

where $score_G^*(C, C^*)$ reflects the average of the maximum match scores for all bi-clusters in C with respect to the bi-clusters in $C*$. In our experiment, $C*$ is one or more reference (optimal) bi-cluster(s) embedded in the expression matrix beforehand. For the parameter settings of the existing algorithms, I follow the previous works [54] and [66].

2.1.2 Results on Synthetic Data.

Because most existing bi-clustering algorithms do not work on linear coherent bi-clusters, I select two bi-clustering algorithms OPSM and ISA that seek additive bi-cluster structures to compare to our LCBD algorithm, since an additive bi-cluster is a special case of a linear coherent bi-cluster. For the MSBE algorithm, I found in our testing that prior knowledge of a reference gene and reference sample for recovering a synthetic bi-cluster had a great effect on its final result, I therefore do not include the MSBE algorithm into the synthetic experiments because it is assumed that this prior knowledge is blind to all the algorithms tested.

Constant bi-cluster. To produce an expression matrix with an additive bi-

cluster, I first randomly generated an 100×50 matrix. The values of the expression matrix obey either a normal distribution (with mean 0 ad SD 1) or a uniform distribution (with minimum 0 and maximum 1), since a real data distribution could be either [11, 48, 22]. Within the expression matrix, I randomly select a row and 10 columns to form a size 10 reference gene vector. I then randomly select 9 other row vectors under the same samples and re-calculate their expression values based on the equation $A(i, J_r) = m_i \times A(i_0, J_r) + b_i$, where $A(i_0, J_r)$ is the reference gene vector, m_i equals to 1, and b_i is a random constant. Random noise is then added to the synthetic bi-cluster: a certain percent of elements in the bi-cluster is randomly selected and replaced with random values which obey the same distribution as the background matrix. I tested noise levels of 0% to 25% with increasing steps of 5%. At each noise level I generated 50 synthetic matrices with bi-clusters and reported a final match score that is the mean over the 50 results. Figure 2.3 shows that our LCBD algorithm obtained the highest match scores for all noise levels and distributions, compared to the two additive bi-cluster type algorithms OPSM and ISA. As one can see, the LCBD algorithm is robust to noise even at noise level 25% to some extend, but not quite. This occurs when a line will be identified by the Hough transform as long as it passes through at least 3 points (samples) and during the majority sample voting. Although the expression value under some samples is destroyed, there are sufficiently many others that their expression values under these samples are not destroyed and thus these samples still obtain more votes than random samples that are not within the linear coherence bi-cluster.

Linear coherent bi-cluster. Because the LCBD algorithm seeks bi-clusters of the linear coherent type, I can then test it directly on the linear coherent bi-clusters. In this experiment, I only use uniform distribution expression matrix, since the normal distribution matrix shows similar results. To generate a linear coherent bi-cluster in a expression matrix, I use the same procedure as in the constant bi-cluster experiment, except that in the equation $A(i, J_r) = m_i \times A(i_0, J_r) + b_i$, the m_i's are no longer 1's but random values. The left part of Figure 2.4 shows the match scores of the LCBD algorithm under different noise levels and different bi-cluster sizes; the right part of Figure 2.4 shows the corresponding gene discovery rates of the

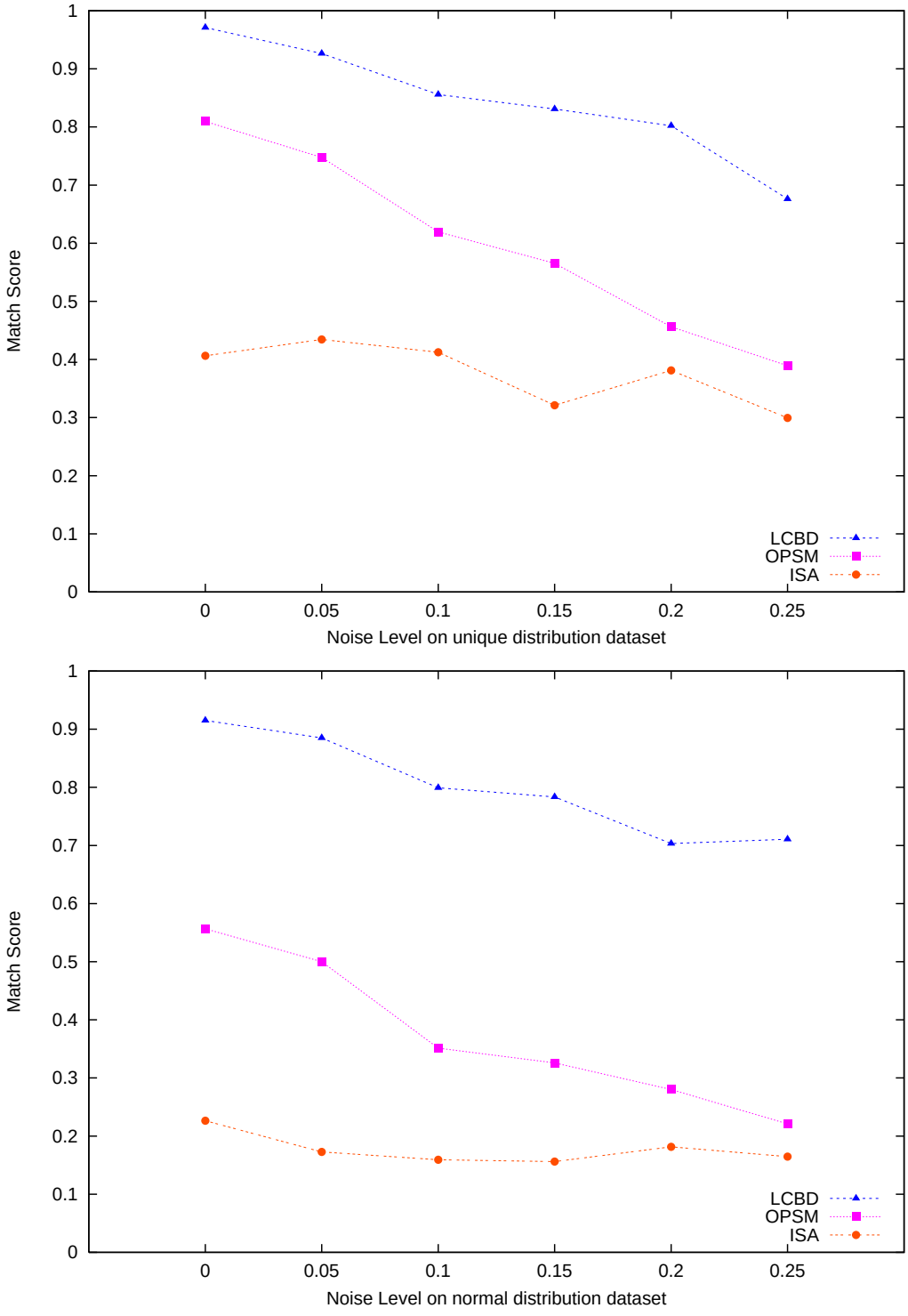


Figure 2.3: Match scores of different additive model bi-clustering algorithms on synthetic dataset under uniform distribution and normal distribution

LCBD algorithm under the same noise level and bi-cluster size. From Figure 2.4, one can see that the match score and gene discovery rates are generally higher on larger bi-clusters. This is the case because whether a line can be identified during the Hough transform depends more on the absolute number of points that a line passes through than the proportion of points a line passes through. This suggests that the LCBD algorithm should be better at discovering large bi-clusters.

Overlapping test. To test the LCBD algorithm on discovering multiple overlapping bi-clusters, I generated two linear coherent bi-clusters in the expression matrix and let them overlap to some degree. Figure A.2 in Appendix A shows the mean match scores of the LCBD algorithm on discovering two overlapping bi-clusters at noise level 10%. For the overlapping elements, I replace their original values with the sum of the two overlapping values. The overlapping elements are not linear coherent elements and can be viewed as noise elements.

2.1.3 Results on Real Data.

The documented descriptions of functions and processes that genes participate in has become widely available prior knowledge. The Gene Ontology Consortium in particular provides one of the largest organized collection of gene annotations. Following the idea in [83, 66], I investigate whether the genes identified in bi-clusters produced by the different algorithms show significant enrichment with respect to a specific Gene Ontology annotation. I use two web-servers, FuncAssociate [5] and EasyGo [96], to evaluate the groups of genes produced in our bi-clustering results. The FuncAssociate computes the hypergeometric functional enrichment score, cf. [5], based on Molecular Function and Biological Process annotations. The resulting scores are adjusted for multiple testing by using the Westfall and Young procedure [87, 5]. The EasyGo calculates the functional enrichment score in a similar way. In detail, based on availability, I tested the bi-clustering results from the *Saccharomyces Cerevisiae* dataset on the FuncAssociate web-server and the results from the *Arabidopsis Thaliana* dataset on the EasyGo web-server. The *Saccharomyces Cerevisiae* dataset contains 2993 genes and 173 conditions and the *Arabidopsis Thaliana* dataset contains 734 genes and 69 conditions. Figure 2.5

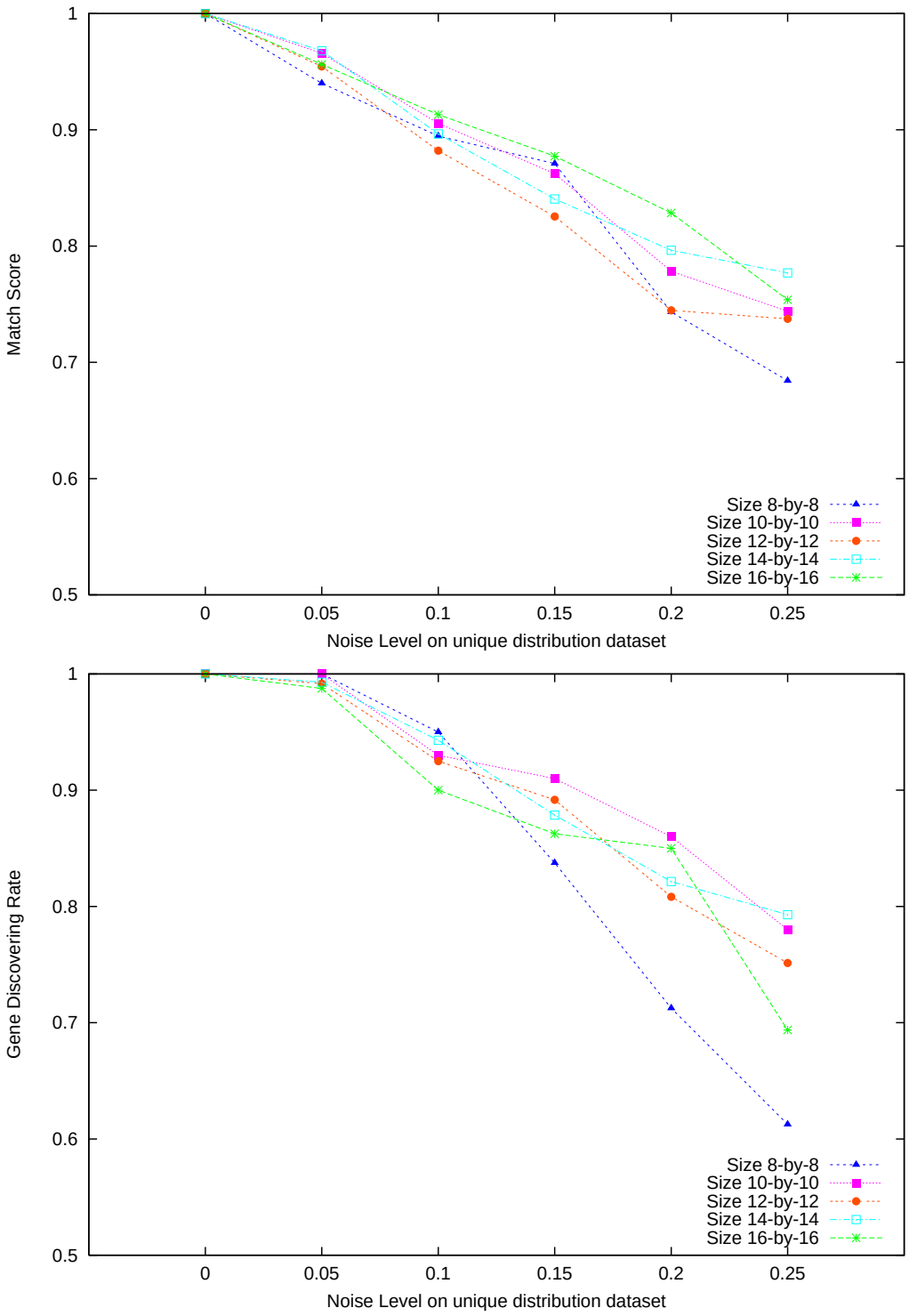


Figure 2.4: Match score and gene discovering rate of the LCBD method on synthetic dataset of different bi-cluster size and noise level under uniform distribution

shows the proportion of gene groups of bi-clusters of S. Cerevisiae and A. Thaliana that are functionally enriched at different significance levels. The LCBD algorithm demonstrates the best results (all 100%) on the *Arabidopsis Thaliana* dataset, compared to other seven algorithms; the LCBD results are also competitive to the best results derived from the MSBE algorithm on the *Saccharomyces Cerevisiae* dataset. These results on real datasets indicate that linear coherent bi-clusters are a useful form of bi-cluster structure to extract from gene expression datasets, and could be a bi-cluster type that exists widely in other gene expression datasets.

2.1.4 Discussion

The experimental results on the synthetic data show that the LCBD algorithm can accurately discover additive and linear coherent bi-clusters, while being robust to the bi-cluster size. The results on the two real datasets revealed that the linear coherent bi-clusters discovered by LCBD are functionally enriched and therefore biologically meaningful. The limitation of the LCBD algorithm is that in the first step, it only collect a line of samples along which a gene pair exhibits linear relation. However, due to unpredictable noise level of microarray data [74, 27], it could be problematic. Figure 2.6 shows that a gene pair's linear correlation could be exhibited by means of a wide beam. Moreover, a pair of genes participates in a linear coherent bi-cluster must be evidenced by a non-trivial subset of samples in which these two genes are co-up-regulated (or co-down-regulated). Therefore, the scatter plot of their pairwise expression levels, see Figure 2.6, where every point (x, y) represents a sample in which the two genes have expression levels x and y respectively, must show a diagonal band with a sufficient number of sample points.

2.2 Approach 2: Beam Detection

To address the two limitations of the LCBD algorithm mentioned in Section 2.1.4, I subsequently proposed the LinCoh algorithm. It starts with composing this non-trivial supporting sample set for each gene pair, then clustering these so-called *outer* sample sets. Each outer sample set cluster, together with the associated genes and

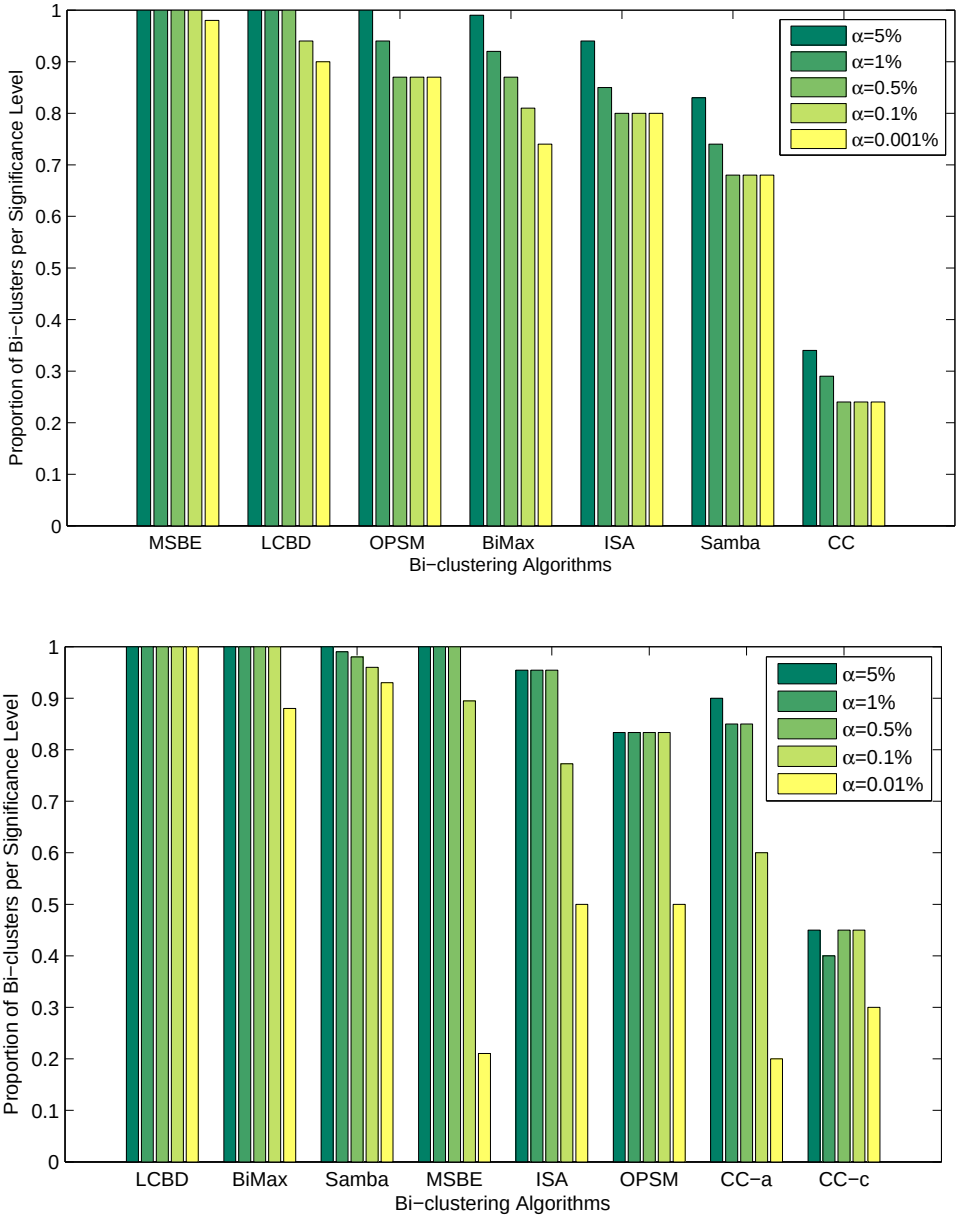
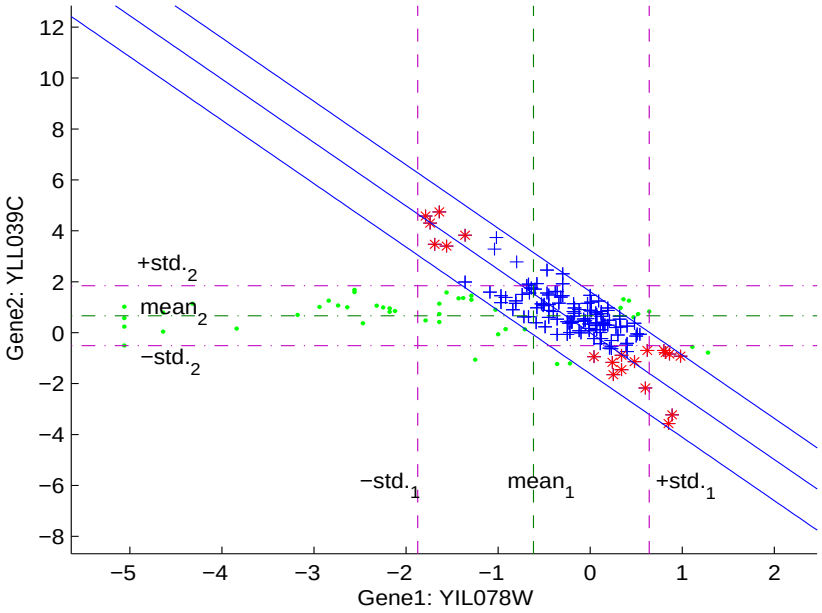
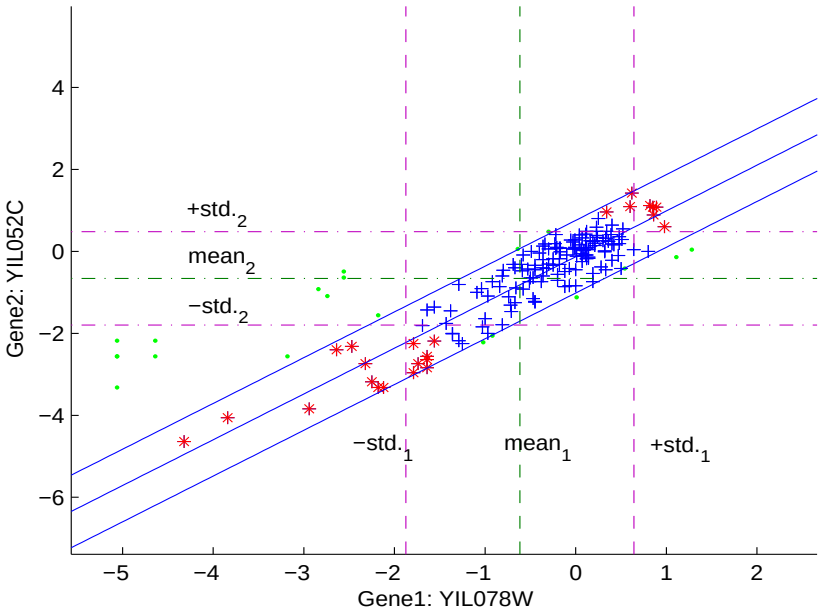


Figure 2.5: Proportion of bi-clusters significantly enriched by any GO biological process category on S. Cerevisiae(left) and A. Thaliana(right) datasets. α is the adjusted significant scores of the bi-clusters



(a) Negative correlation.



(b) Positive correlation.

Figure 2.6: (a) illustrates two yeast genes *YIL078W* and *YLL039C* that have negative expression correlation under a subset of conditions; the red conditions provide a stronger evidence than the blue conditions, whereas the green conditions do not suggest any correlation. Similarly in (b), genes *YIL078W* and *YIL052C* show a positive expression correlation.

inner samples, is filtered to produce a final bi-cluster. This work was published in [74].

2.2.1 Method

The LinCoh algorithm is composed of two major steps. In the first step, we discover the gene pairwise linear relations via a beam detection process, that, compared to LCBD, is more robust to noise and discover samples under which gene pair is co-expressed. In the second step, bi-clusters are constructed based on sample set clustering, which we believe is more effective than the majority voting used in LCBD for finding common sample sets under which a group of genes are co-expressed. A detailed description of the LinCoh algorithm can be found in Appendix B.

2.2.2 Results on Synthetic Data

I investigated the LinCoh algorithm, and again made comparisons with the five previously mentioned bi-clustering algorithms, LCBD, CC, OPSM, ISA, and MSBE, initially on synthetic datasets. On synthetic datasets, bi-clustering algorithms are evaluated on their ability to recover the implanted (true) bi-clusters. Prelić's gene match score and overall match score [66] are again adopted. Let $\mathcal{C}$ and $\mathcal{C}^*$ denote the set of output bi-clusters from an algorithm and the set of true bi-clusters for a dataset. The gene match score of $\mathcal{C}$ with respect to the target $\mathcal{C}^*$ is defined as $\text{score}_G(\mathcal{C}, \mathcal{C}^*) = \frac{1}{|\mathcal{C}|} \sum_{(G_1,S_1)\in\mathcal{C}} \max_{(G_1^*,S_1^*)\in\mathcal{C}^*} \frac{|G_1 \cap G_1^*|}{|G_1 \cup G_1^*|}$, which is essentially the average of the maximum gene match scores of bi-clusters in $\mathcal{C}$ with respect to the target bi-clusters. Similarly, the sample match score $\text{score}_S(\mathcal{C}, \mathcal{C}^*)$ can be defined by replacing gene sets with the corresponding sample sets in the above. The overall match score is then defined as their geometric mean, *i.e.*

$$\text{score}(\mathcal{C}, \mathcal{C}^*) = \sqrt{\text{score}_G(\mathcal{C}, \mathcal{C}^*) \times \text{score}_S(\mathcal{C}, \mathcal{C}^*)}. \tag{2.2}$$

Noise resistance test. This experiment examines how well a bi-clustering algorithm can recover implanted bi-clusters. I follow Prelić's testing strategy to first generate a 100×50 background matrix (*i.e.*, 100 genes and 50 samples), using a standard normal distribution for the matrix elements; I then embed ten

10×5 non-overlapping linear coherent bi-clusters along the diagonal; later for each vector of the five expression values, the first two of them are set to down-regulated, the last two are set to up-regulated, and the middle one is non-regulated; lastly, I add noise to the embedded bi-clusters at six different noise levels ($\ell = 0.00, 0.05, 0.10, 0.15, 0.20, 0.25$) by perturbing the entry values so that the resultant values are ℓ away from the original values. The generation is repeated ten times to give ten matrices. The same simulation process is done to generate synthetic datasets containing additive bi-clusters, when I compare the bi-clustering algorithms on their performance to recover additive bi-clusters only (which is a special case of linear coherent bi-clusters). Note that the linear coherence of the bi-clusters we generated now is near the positive or negative diagonals of the gene pair 2D plot, which is different from the linear coherence of the synthetic bi-clusters we generated for testing the LCBD algorithm where the linear coherence can be anywhere on the gene pair 2D plot. Another difference compared to the synthetic data generation in LCBD is that 5 samples rather than 10 samples are used for each bi-cluster. We do so by considering two reasons: first, to embed exactly 10 bi-clusters in each background matrix; second, to make the bi-clusters more challenging for different algorithms to discover because linear coherence evidenced by less samples are harder to discover.

Figure 2.9 shows the gene match scores of all six bi-clustering algorithms at six different noise levels, on their performance of recovering linear coherent bi-clusters and additive bi-clusters, respectively. Their overall match scores and gene discovery rates (defined as the percentage of genes in the output bi-clusters over all the genes in the true bi-clusters) can be found in Figures 2.7 and 2.8. In terms of match scores, Figures 2.9 and 2.7 clearly show that our LinCoh outperformed all the other four algorithms, ISA ranked the second, and the other four performed quite poorly. LCBD performs well on low noise level but quickly drops when noise level increases. In terms of gene discovery rate, again LinCoh outperformed all the other four algorithms. I remark that gene discovery rate can be trivially lifted up by simply output more bi-clusters. It is not a main measure used in this work, but a useful measure in conjunction with match scores.

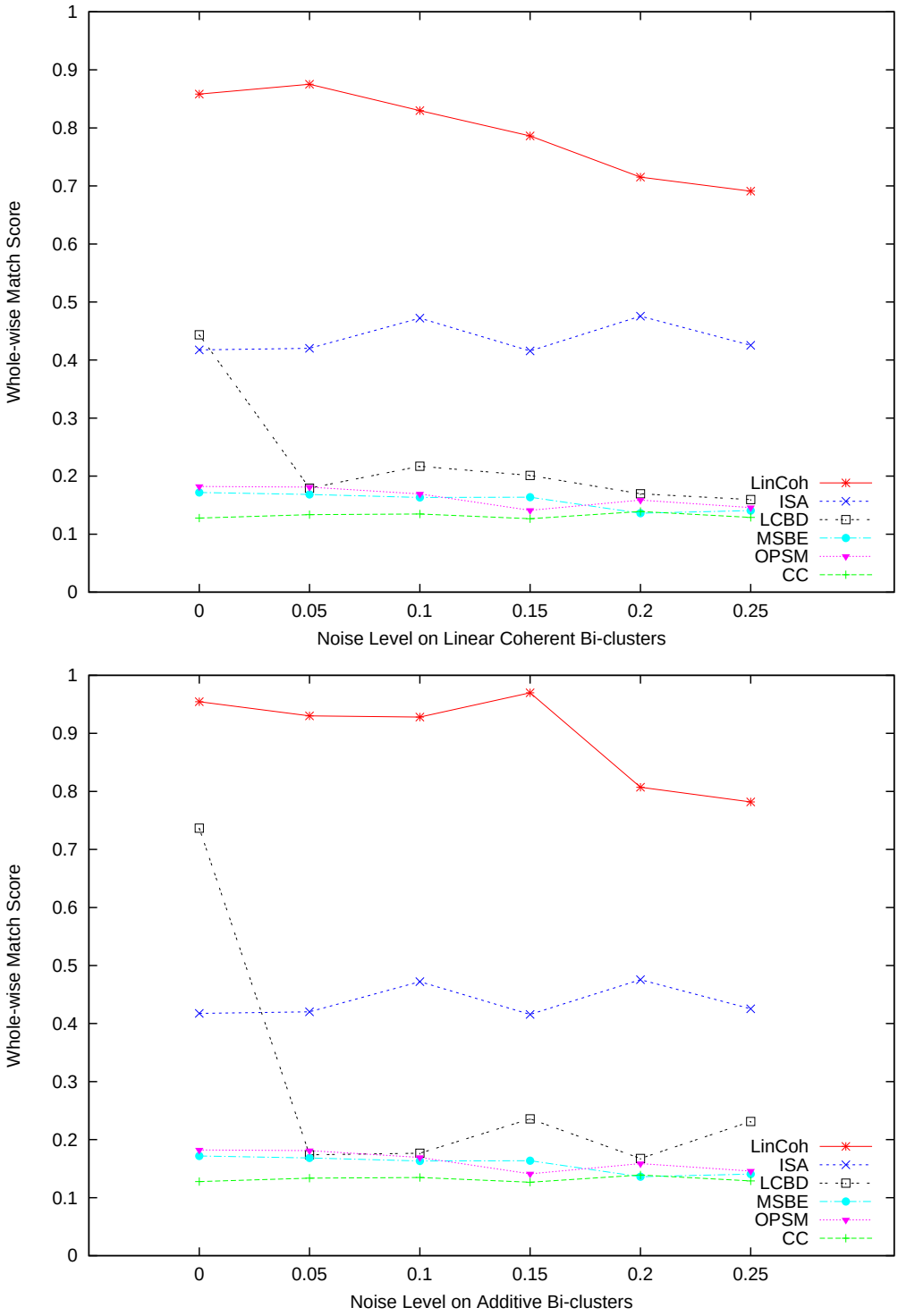


Figure 2.7: The overall match scores of the five algorithms for recovering linear coherent and additive bi-clusters, at six different noise levels.

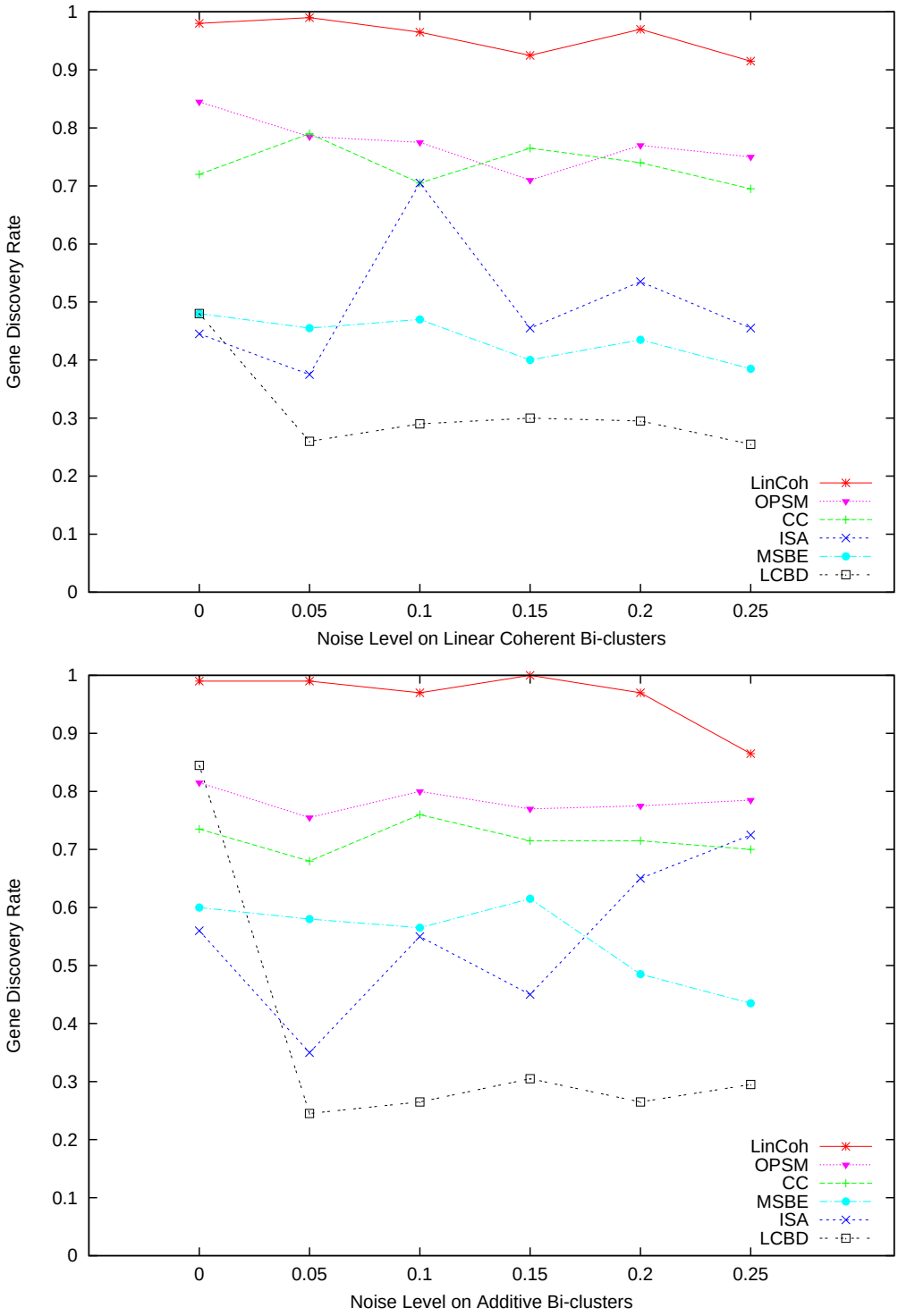


Figure 2.8: The gene discovery rates of the five algorithms for recovering linear coherent and additive bi-clusters, at six different noise levels.

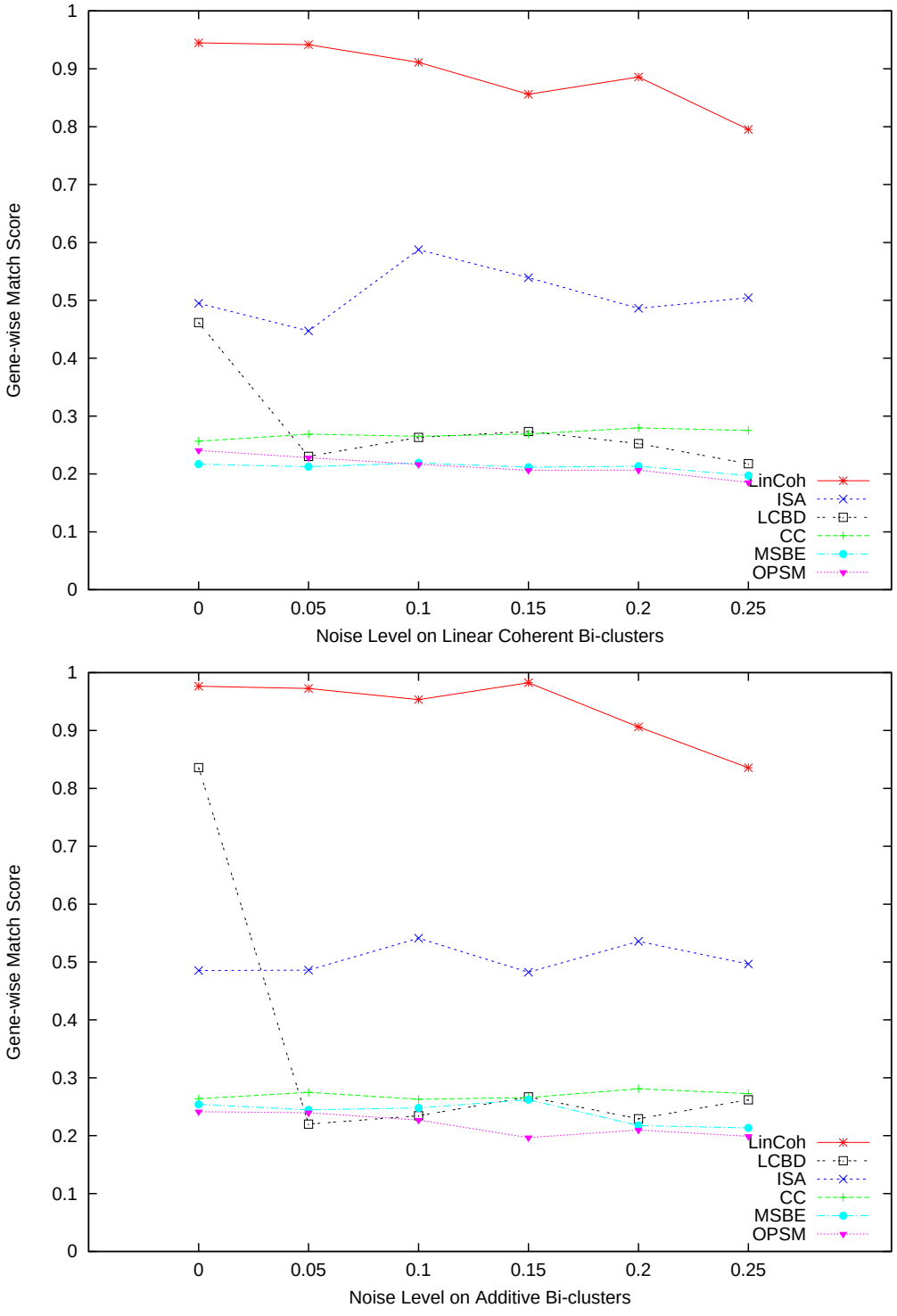


Figure 2.9: The gene match scores of the five algorithms on recovering linear coherent bi-clusters and additive bi-clusters at six different noise levels.

Overlapping test. Individual genes can participate in multiple biological processes, yielding bi-clusters that overlap with common genes in an expression matrix. Bi-clusters might also overlap with a subset of samples. This experiment is designed to examine the ability of different bi-clustering algorithms to recover overlapping bi-clusters. As before, I consider type-(f) linear coherent bi-clusters and type-(d) additive bi-clusters, at a fixed noise level of $\ell = 0.1$. Again, ten 100×50 background matrices are generated using a standard normal distribution for the matrix elements; into each of them, two 10×10 bi-clusters are embedded, overlapping with each other by one of the following six cases: 0×0, 1×1, 2×2, 3×3, 4×4, and 5×5. Previous simulation studies suggested to replace the matrix elements in the overlapped area with a random value; I expect, however, these overlapping genes to obey a reasonable logic such as the AND gate and the OR gate leading to a *union* and an *additive* behavior. Therefore, in the union overlap model, the matrix entries in the overlapped area preserve linear coherency in both bi-clusters (consequently, the overlapped area extends its linear coherency into both bi-clusters on those samples in the overlapped area); and in the additive overlap model, these entries take the sum of the gene expression levels from both bi-clusters.

Figure B.1 in Appendix B shows the gene match scores of the five bi-clustering algorithms in this experiment. Their overall match scores and gene discovery rates under the union overlap model are plotted in Figures B.2 and B.3 in Appendix B. The results of the additive overlap model are in Figures B.4, B.5, and B.6 in Appendix B. From all these results, one can see that our LinCoh outperformed the other four algorithms; OPSM and MSBE performed worse, but similarly to each other; CC performed the worst; and ISA demonstrated varying performance. LCBD again shows better performance on low noise level and worse performance on high noise level. Note that he overall performance of LCBD on the overlapping test is better than on the noise test because the bi-clusters in the overlapping test has bigger sample size which makes the Hough transform easier to capture the lines.

2.2.3 Results on Real Data

On real datasets, due to dataset availability, rather than using the servers FuncAssociate and EasyGo in LCBD, the bi-clusters discovered by an algorithm are mapped to known biological pathways, defined in the more GO functional classification scheme [1], the KEGG pathways [41], the MIPS yeast functional categories [68] (for yeast dataset), and the EcoCyc database [46] (for E.coli dataset), to obtain their *gene functional enrichment score* as implemented in [53]. The average absolute correlation coefficients (aacc's) of the discovered bi-clusters are also used to compare different algorithms.

Two datasets Yeast and E.coli are used in the real dataset experiments and the Arabidopsis Thaliana dataset used in LCBD is not used here due to its small number of samples. The yeast dataset is obtained from [23], containing 2993 genes on 173 samples; the E.coli dataset (version 4 built 3) is from [21], which contains initially 4217 genes on 264 samples. Genes with small expression deviations were removed from the second dataset, giving rise to 3016 genes. Such a process ensures that all five bi-clustering algorithms can run on the dataset. In particular, it took two weeks for LinCoh to run on each dataset using a 2.2GHz CPU node of 2.5GB memory. The performance of an algorithm on these two real datasets is measured in gene functional enrichment score [53]. First, the P-value of each output bi-cluster is defined using its most enriched functional class (biological process). The probability of having r genes of the same functional class in a bi-cluster of size n from a genome with a total of N genes can be computed using the hypergeometric function, where p is the percentage of that functional class of genes over all functional classes of genes encoded in the whole genome. Numerically [53],

$$Pr(r|N,p,n) = \binom{pN}{r} \cdot \binom{(1-p)N}{n-r} / \binom{N}{n} \tag{2.3}$$

Such a probability is taken as the P-value of the output bi-cluster enriched with genes from that functional class [53]. The smallest P-value over all functional classes is defined as the P-value of the output bi-cluster — the smaller the P-value of a bi-cluster the more likely its genes come from the same biological process. For each algorithm, I calculate the fraction of its output bi-clusters whose P-values are

smaller than a significance cutoff α.

Figure 2.10 compares the six algorithms using six different P-value cutoffs, evaluated on the GO database. Results on the KEGG, MIPS, and Regulon databases are in Figures 2.11 and 2.12. All these results show that our LinCoh consistently performed well; OPSM and ISA did not perform consistently on the two datasets across databases; and that MSBE and CC did not perform as well as the other three algorithms. The LCBD algorithm consistently performs worse than LinCoh under these two datasets and these experimental settings. This is mainly because the linear coherent relation it considers is not always close to the main diagonal its less robust to noise.

One potential issue with the P-value based performance measurement is that P-values are sensitive to the bi-cluster size [53]; in general, larger bi-clusters tend to lead to smaller P-values (more significant). Table B.1 in Appendix B summarizes the statistics of the bi-clusters produced by the five algorithms. The last column in the table records the numbers of unique functional terms enriched by the produced bi-clusters. On yeast dataset, when measured by the gene enrichment significance test, OPSM performed very well (Figure 2.10, left); yet its bi-clusters only cover one functional term on the GO and KEGG databases and two terms on MIPS database. Such a phenomenon suggests that its bi-clustering result is biased to a group of correlated genes, missed by the P-value based significance test. Furthermore, I generated all the gene pairs with absolute correlation coefficient greater than or equal to 0.8 over all the samples for both the yeast and E.coli datasets. Table B.2 in Appendix B shows the numbers of common GO terms and their counts. Among these strongly correlated gene pairs, many do not even have one common GO term. Table B.2 in Appendix B shows the top 10 counted common GO terms (the full table can be found at '`http://www.cs.ualberta.ca/~ys3/LinCoh`').

The above two potential issues hint that the P-value based evaluation is meaningful but has limitations. I propose to use the average absolute correlation coefficient over all gene pairs in a bi-cluster defined in Eq. (B.1) as an alternative assessment of the quality of a linear coherent bi-cluster. Figure 2.15 shows the box plot of these correlation values for the bi-clusters produced by the five algorithms

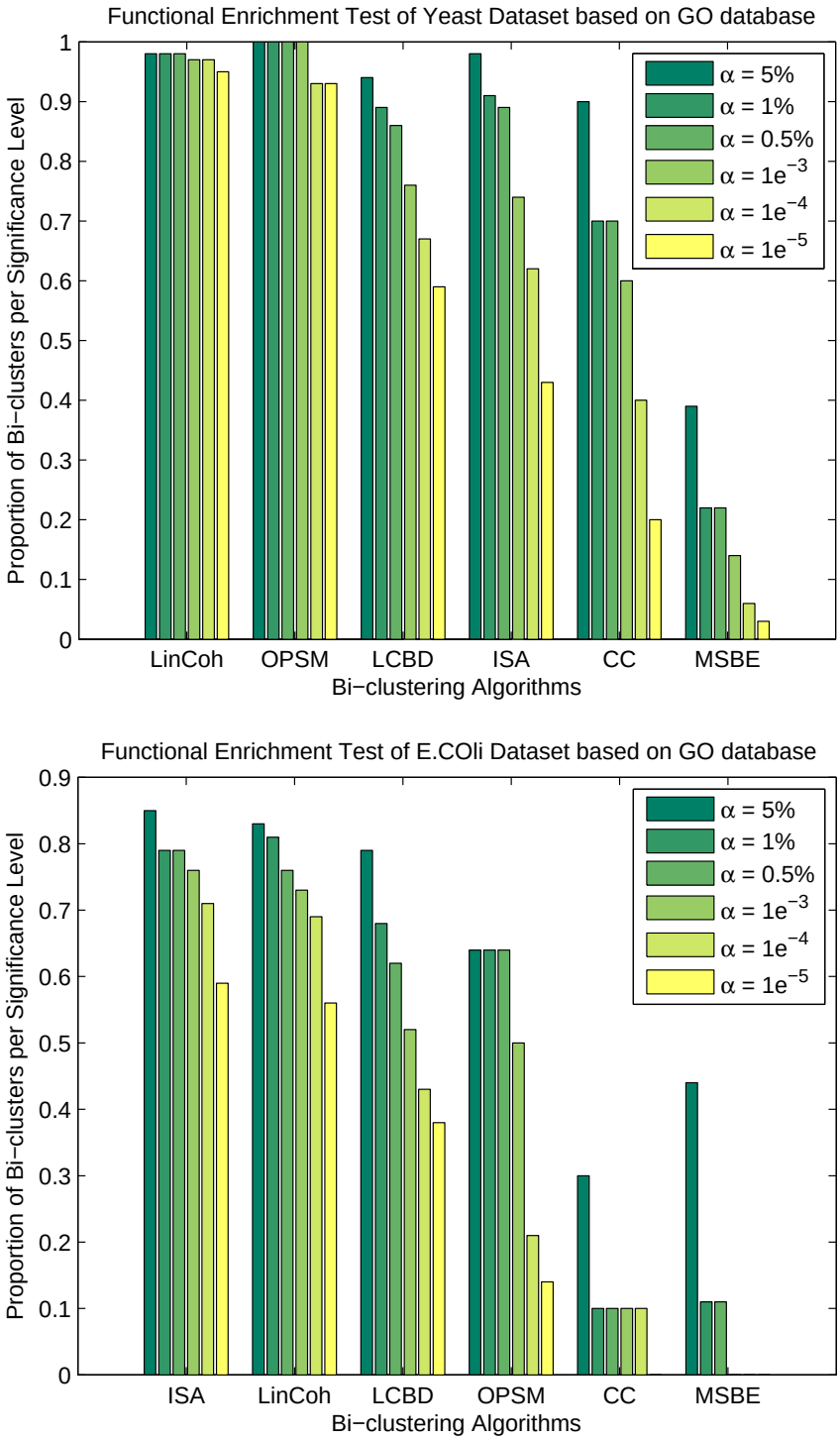


Figure 2.10: Proportion of discovered bi-clusters by the five algorithms on the two real datasets that are significantly enriched the GO biological process, using six different P-value cutoffs.

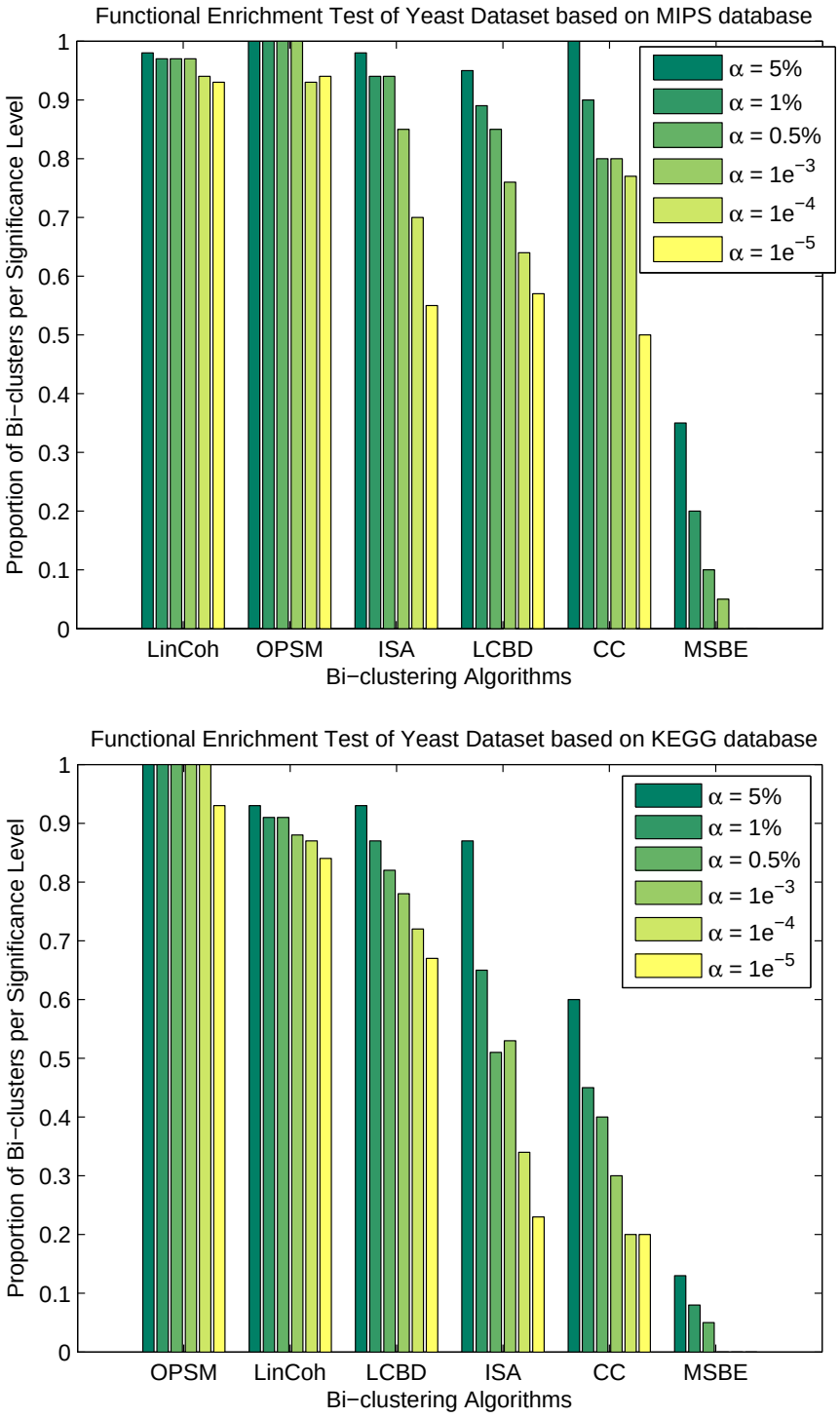


Figure 2.11: Proportion of yeast bi-clusters that are significantly enriched over different P-values in the MIPS pathway and KEGG pathway.

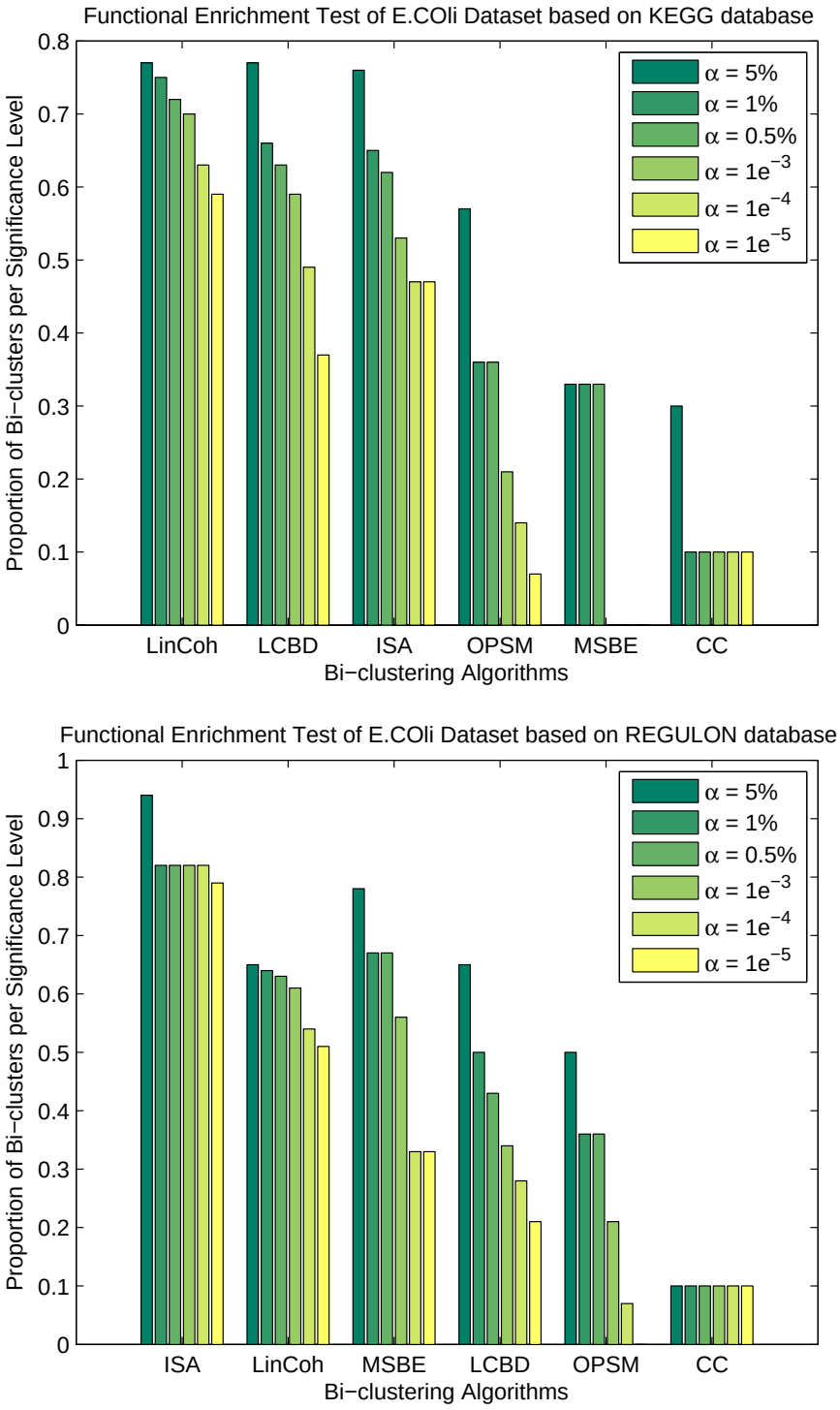


Figure 2.12: Proportion of E.coli bi-clusters that are significantly enriched over different P-values in the KEGG pathway and experimentally verified regulons.

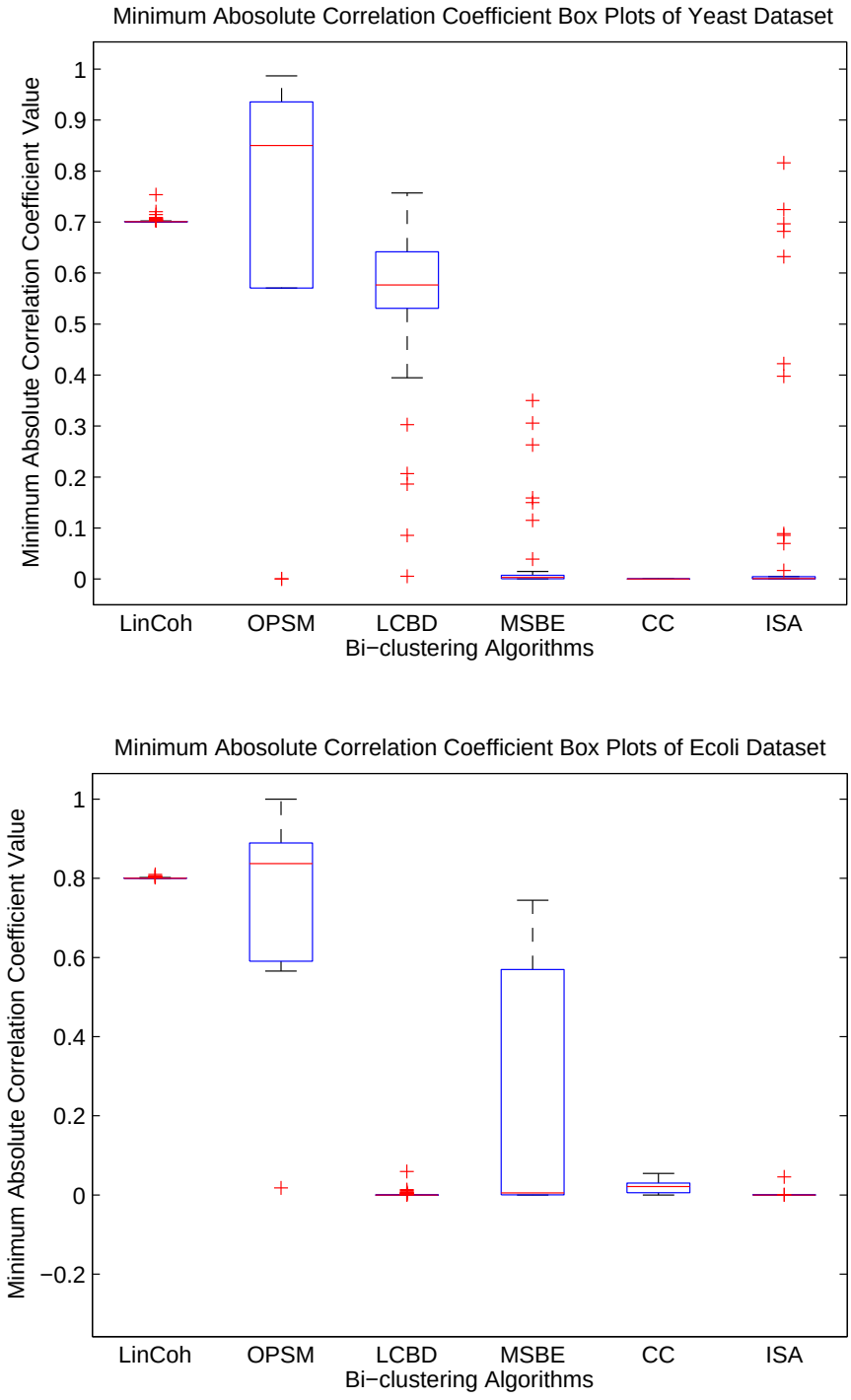
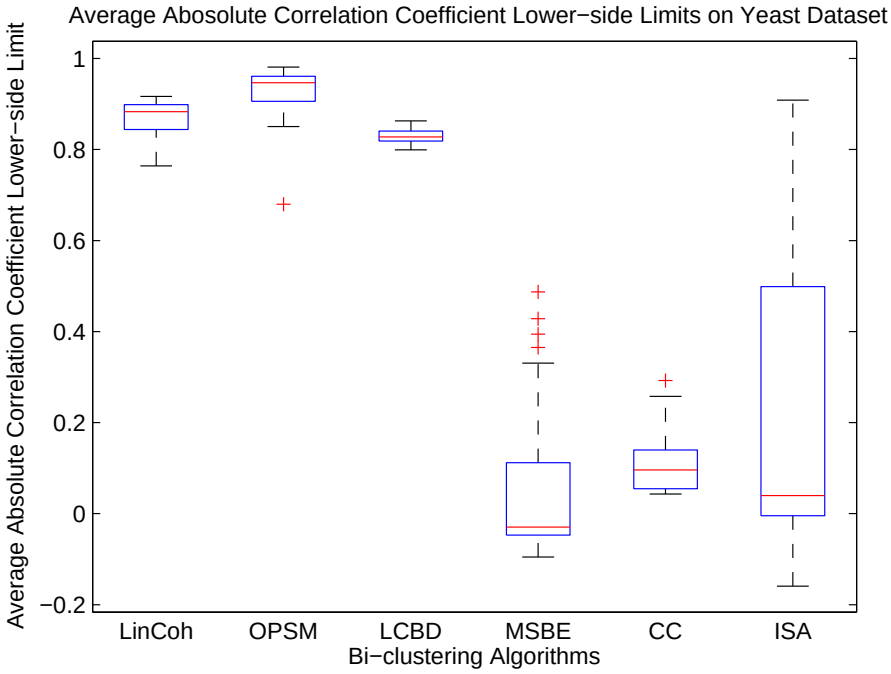


Figure 2.13: The box plots of minimum absolute correlation coefficients of the bi-clusters produced by the five algorithms on the yeast and E.coli datasets.



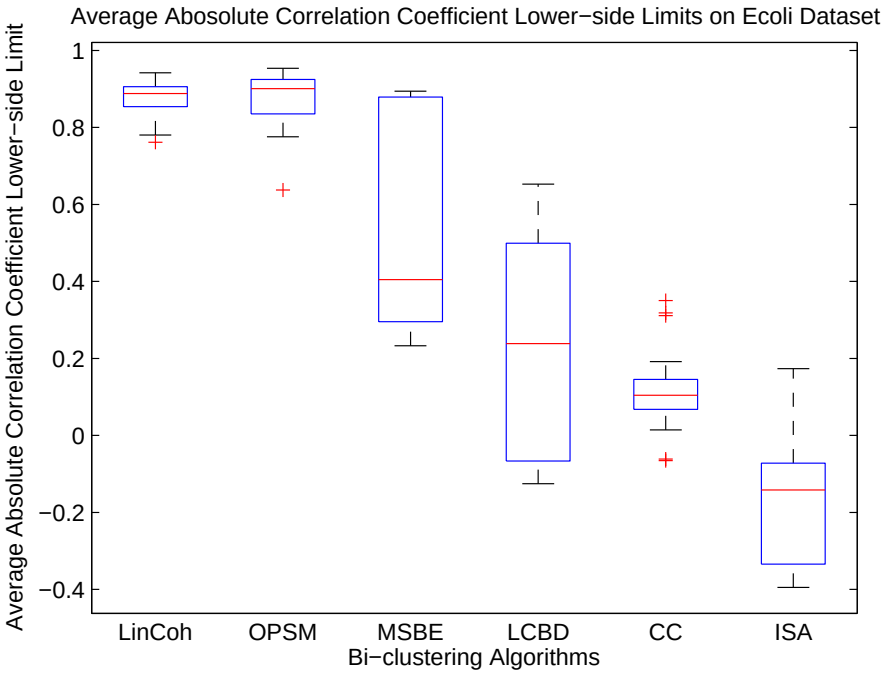


Figure 2.14: The box plots of the 99% confidence thresholds of the average absolute correlation coefficients of the bi-clusters, using the number of samples in each bi-cluster, produced by the five algorithms on the yeast and E.coli datasets.

on the two real datasets. From the figure, one can see that our LinCoh and OPSM significantly outperformed the other three algorithms. Additionally, the minimum absolute correlation coefficient over all gene pairs in a bi-cluster can also be adopted as a quality measurement. Figure 2.13 shows these results.

Figure 2.15 shows that OPSM produced bi-clusters with very high linear coherence. But the numbers of samples in its bi-clusters are much smaller than those in LinCoh's bi-clusters, as shown in Table B.1 in Appendix B (tens versus hundreds). This suggests that very closely interacting gene pairs can have small empirical correlation coefficients on a subset of samples, largely due to noise and measurement errors. In fact, there is always a trade-off between bi-cluster coherence and its size. Thus, to compare algorithms in a less sample-size biased way, I replaced for each bi-cluster its average absolute correlation coefficient by the 99% confidence threshold using the number of samples in the bi-cluster [71], and box plotted these values in Figure 2.14. They show much more comparable performance between LinCoh and OPSM.

2.2.4 Discussion

LCBD and LinCoh both identify linear-coherent bi-clusters that disclose undirected relations between genes, while LinCoh is a more practical algorithm on microarray datasets due to data noise. The experiments on synthetic and real datasets demonstrate that LinCoh consistently performed competitively with other bi-clustering algorithms. On real datasets, I showed some limitations of the widely adopted functional enrichment measurement, and proposed to use average absolute correlation coefficient as an alternative measure for bi-clustering quality. Given its outperformance of existing popular algorithms, LinCoh can serve as another useful tool for microarray data analysis, including bi-clustering and genetic regulatory network inference. A disadvantage of LinCoh are its large memory and compute time requirements, due to constructing the outer and inner sample set matrices. It takes $O(n^2 m_p)$ to compute the sample set matrices where n is the number of genes and m_p is the number of parameters θ, β and γ. The memory required for storing the matrices is $O(n^2 m_s)$ where n is the number of genes and m_s is the average size

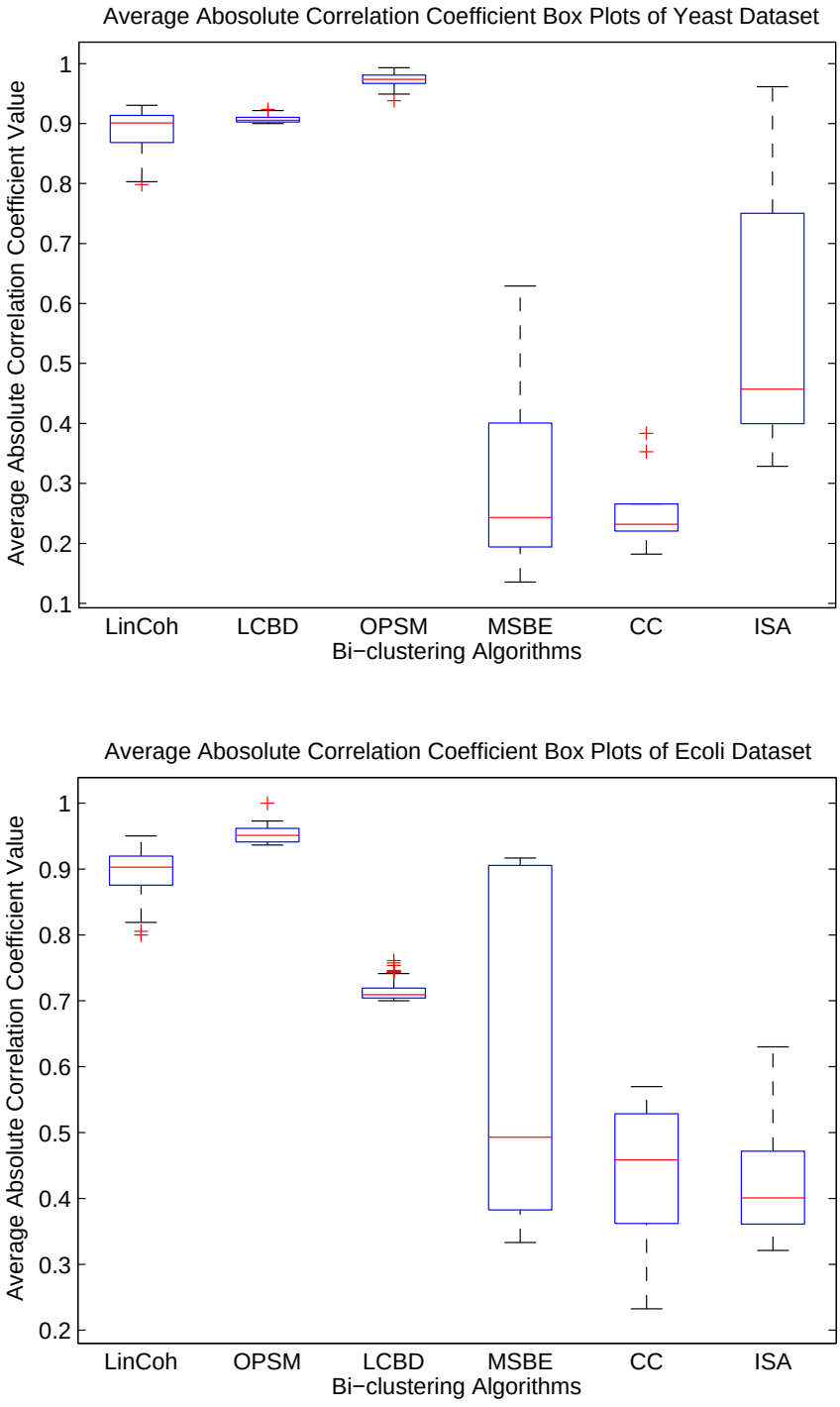


Figure 2.15: Box plots of the average absolute correlation coefficients obtained by the five bi-clustering algorithms on yeast and E.coli datasets, respectively.

of the sample set elements. It takes weeks and up to 1 Gigabyte memory to run experiments on the E.coli dataset. Improvements in beam detection and sample set clustering can also achieve significant speed-ups.

2.3 Approach 3: Sparse Learning

After introducing the above two heuristic approaches, I investigated the linear coherent bi-clustering problem from a sparse learning perspective. In doing so I exploited the experience I gained in modeling with sparse machine learning methods in Chapter 3 below. The goal is to re-express the previous heuristic approaches as a simple loss minimization problem, using sparse regularization to automatically perform gene-sample selection. The advantage of such an approach is that it decouples the principle by which bi-clusters are identified from the algorithmic techniques used to solve the optimization problem. In this way we can exploit state of the art optimization tools and can concentrate our efforts on formulating good objectives.

2.3.1 Method

Let us first consider a pair of $1 \times p$ observation vectors $\mathbf{m_{i:}}$ and $\mathbf{m_{j:}}$. Here $\mathbf{m_{i:}}$ is defined as the ith row vector of matrix M. Other row/column vectors appearing later in this section will be written in the same way. For a given subset of features we can always find the linear regression of this pair of observations in a 2D space that gives us least sum of residuals. We denote the linear regression by slope a_{ij} and intercept b_{ij}. Now the problem is to select a subset of feature so that the sum of residuals from the best regression is minimized. For the bi-cluster that is generated based on the ith observation, we introduce a $1 \times p$ feature selection vector $\mathbf{s_i} \in \{0, 1\}$, where $s_{ik} = 1$ if the kth feature is selected and 0 otherwise. Without any constraint, this problem will always give a trivial solution $\mathbf{s_i} = \mathbf{0}$, yielding a zero sum of residuals. Therefore, we add a regularizer $\beta_1 \|\mathbf{1} - \mathbf{s_i}\|_1$ to penalize any solution with too few $s_{ik} = 1$ values, where $\mathbf{1}$ denotes a vector of all 1s. In the subsequent formulations we choose the L_1 norm because it gives us a sparse solution in $\mathbf{1} - \mathbf{s_i}$ once $\mathbf{s_i}$ has been relaxed to $[0, 1]$. For a single row i, the problem can then be formulated as an

optimization as follows:

$$\begin{aligned} \min_{\mathbf{s_i}, a_{ij}, b_{ij}} & \sum_k s_{ik}(m_{ik} - a_{ij}x_{ik} - b_{ij})^2 + \beta_1 \|\mathbf{1} - \mathbf{s_i}\|_1 \\ s.t. \quad & s_{ik} \in \{0, 1\} \end{aligned} \tag{2.4}$$

Now, consider the whole matrix M from which we want to detect a set of row-wise linear coherent bi-clusters. We introduce a $n \times n$ binary matrix W, where $w_{ij} = 1$ indicates there is strong linear coherence between the observation pair $(i,\ j)$ and $w_{ij} = 0$ otherwise. By extending 2.4 in terms of the whole matrices M, S, A, B and introducing W, we obtain the complete formulation:

$$\begin{aligned} \min_{W,S,A,B} & \sum_{i,j} w_{ij} \sum_k s_{ik}(m_{ik} - a_{ij}x_{ik} - b_{ij})^2 \\ & + \beta_1 \|\mathbf{1} \cdot \mathbf{1}^T - S\|_{1,1} + \beta_2 \|\mathbf{1} \cdot \mathbf{1}^T - W\|_{1,1} \\ s.t. \quad & w_{ij} \in \{0, 1\}, s_{ik} \in \{0, 1\} \end{aligned} \tag{2.5}$$

where W can be interpreted as observation (data) selection matrix, and S can be interpreted as the feature (sample) selection matrix. Here S is a $n \times p$ binary matrix with the ith row corresponding to the feature selection vector for the ith observation. Note that the sparse regularizer $\beta_1 \|\mathbf{1} - \mathbf{s_i}\|_1$ becomes $\beta_1 \|\mathbf{1} \cdot \mathbf{1}^T - S\|_{1,1}$. Similarly, we add another sparse regularizer $\beta_1 \|\mathbf{1} \cdot \mathbf{1}^T - W\|_{1,1}$ to penalize trivial solutions where W is set too close to the identity matrix. We want to favor the case that the scatter points (feature points) of a pairwise 2D plot do not stick together so that to exhibit better linear coherence. Considering this, we introduce a $n \times n \times p$ matrix D, where $d_{ijk} \in [0, 1]$ indicates the importance of the kth feature under the observation pair (i, j). In the gene expression matrix case, because it is desired to favor co-up-regulated and co-down-regulated gene expression samples, we assign $d_{ijk} = e^{d'_{ijk}}$, where d'_{ijk} is the Euclidean distance of the kth data point to the central point $(\bar{m}_{i:}, \bar{m}_{j:})$. Different prior knowledge can be introduced to form D from other data sources. Therefore, after relaxing $W \in \{0, 1\}$ to $W \in [0, 1]$ and $S \in \{0, 1\}$ to $S \in [0, 1]$, we get:

$$\begin{aligned} \min_{W,S,A,B} & \sum_{i,j} w_{ij} \sum_k s_{ik} \frac{1}{d_{ijk}} (m_{ik} - a_{ij}x_{ik} - b_{ij})^2 \\ & + \beta_1 \|\mathbf{1} \cdot \mathbf{1}^T - S\|_{1,1} + \beta_2 \|\mathbf{1} \cdot \mathbf{1}^T - W\|_{1,1} \\ s.t. \quad & w_{ij} \in [0, 1], s_{ik} \in [0, 1] \end{aligned} \tag{2.6}$$

By introducing some new notation, we can re-express this problem in an equivalent form that proves to be more convenient for formulating an efficient iterative procedure below. Let $\otimes$ denote Kronecker product, let $\triangle(\mathbf{m})$ denote putting a vector $\mathbf{m}$ on the main diagonal of a square matrix, and let $\div$ denote component-wise divide. Then 2.6 can be equivalently re-written in terms of $\mathbf{s_{i:}}$ and $\mathbf{w_{i:}}$ as:

$$\begin{aligned}\min_{W,S,A,B} \sum_i \left\| \triangle(\mathbf{w_{i:}})^{1/2}(\mathbf{1} \otimes \mathbf{m_{i:}} - \triangle(\mathbf{a_{i:}})M - \triangle(\mathbf{b_{j:}})\mathbf{1} \otimes \mathbf{1}^T) \div D_i^* \triangle(\mathbf{s_{i:}})^{1/2} \right\|_F^2 \\ +\beta_1 \|\mathbf{1} \cdot \mathbf{1}^T - S\|_{1,1} + \beta_2 \|\mathbf{1} \cdot \mathbf{1}^T - W\|_{1,1} \\ s.t. \quad w_{ij} \in [0,1], s_{ik} \in [0,1]\end{aligned} \tag{2.7}$$

where D_i^* has the same dimension as D_i with each element equals to the square root of the corresponding element in D_i.

Unfortunately, 2.7 is not jointly convex in W, S, A and B, so we are currently unable to formulate an efficient global optimization procedure. Nevertheless, an efficient iterative procedure can be devised that finds a reasonable local solution.

Initialization: Because of the potential difficulty of local minima, initialization of W, S, A, and B becomes very important for solving (2.7) iteratively. To simplify the initialization, and allow a generally effective approach, we first normalize the data matrix M so that each row $\mathbf{m_{i:}} \in [0,1]$. In the case of gene expression analysis, A is initialized to $\mathbf{1} \cdot \mathbf{1}^T$ since a gene pair that has strong correlation will have a sufficient number of samples under which the gene pair has a co-up-regulated and co-down-regulated pattern, which implies that on normalized data, the slope is near 1. The intercept b_{ij} is normalized in a way that the linear regression line for each observation pair passes through the central point $(\bar{m}_{i:}, \bar{m}_{j:})$ with slope a_{ij}. After A and B is initialized, $\mathbf{s_{i:}}$ is initialized such that $s_{ik} = 1$ if the distance d'_{ijk} of kth data point of the (i,j) pair to the line (a_{ij}, b_{ij}) is within some threshold. Since the data is normalized, an appropriate threshold can be set for data of the same type and will not affect the results to a large extent. In the case of gene expression data, since we want to favor sample points that are far away from the central point $(\bar{m}_{i:}, \bar{m}_{j:})$, we set the threshold as a monotonically increasing function of the distance d''_{ijk} between $(\bar{m}_{i:}, \bar{m}_{j:})$ and the point (m_{ik}, m_{jk})'s projected point on the regression line. We do not initialize W as it will be immediately determined from the initial S, A, and B.

Iterative update of W, S, A and B:

Updating W. Denote the objective function in (2.7) by $f(W, S, A, B)$. Assume that S, A, and B are fixed (initialized as mentioned above for the first iteration). Then the objective function is a convex (linear) function of W and we can optimize W element by element in closed form. In particular, for each w_{ij},, by ignoring constant terms, the problem is equivalent to minimizing $f(w_{ij})$:

$$\begin{aligned} \min_{w_{ij}} & \; w_{ij} \left(\sum_k \frac{s_{ik}}{d_{ijk}} (m_{ik} - a_{ij} m_{jk} - b_{ij})^2 - \beta_2 \right) \\ s.t. & \quad w_{ij} \in [0, 1] \end{aligned} \tag{2.8}$$

Because $f(w_{ij})$ is a linear function of w_{ij}, we obtain $w_{ij} = 1$ if $\sum_k \frac{s_{ik}}{d_{ijk}} (m_{ik} - a_{ij} m_{jk} - b_{ij})^2 < \beta_2$ and $w_{ij} = 0$ otherwise.

Updating S. When W, A, and B are fixed, $f(W, S, A, B)$ becomes a convex (linear) function of S, so similar to updating W, we can update S element by element in closed form. In this case, s_{ik} can be calculated by minimizing $f(s_{ik})$ as follows:

$$\begin{aligned} \min_{s_{ik}} & \; s_{ik} \left(\sum_j \frac{w_{ij}}{d_{ijk}} (m_{ik} - a_{ij} m_{jk} - b_{ij})^2 - \beta_1 \right) \\ s.t. & \quad s_{ik} \in [0, 1] \end{aligned} \tag{2.9}$$

Hence, $s_{ik} = 1$ if $\sum_j \frac{w_{ij}}{d_{ijk}} (m_{ik} - a_{ij} m_{jk} - b_{ij})^2 < \beta_1$ and $s_{ik} = 0$ otherwise.

Updating A and B. When W and S are fixed, the minimization over A and B becomes a standard least squares linear regression problem for each observation pair. In particular, we have:

$$(a_{ij}, b_{ij})^T = (X_{ij}^T \triangle(\mathbf{s_{i:}} \bullet \mathbf{d_{ij:}}) X_{ij})^{-1} X_{ij}^T \triangle(\mathbf{s_{i:}} \bullet \mathbf{d_{ij:}}) \mathbf{y_{ij}} \tag{2.10}$$

where $\bullet$ denotes inner product, $x_{ij} = (\mathbf{1}, \mathbf{m_{j:}}^T)$, and $\mathbf{y} = \mathbf{m}(\mathbf{i}, :)'$

Finally, each of W, S, A, and B are iteratively updated until the objective function converges. Algorithm 3 in Appendix C gives the details of the SLLB algorithm. Note that the time complexity of the SLLB for one iteration is $O(n^2)$. Later experiments on synthetic datasets shows that SLLB converges after 6-8 iterations, which takes less than 10 seconds. On real datasets, good results can be obtained after 10-20 iterations, which take tens of hours.

2.3.2 Results on Synthetic Data

I first compare SLLB with seven existing representative bi-clustering algorithms—QUBIC, LinCoh, LCBD, CC, OPSM, ISA, and MSBE—on synthetic datasets. The QUBIC method I add here is a recent bi-clustering algorithm, which finds bi-clusters by a combination of (semi-) qualitative measures of gene expression data and a combinatorial optimization technique [53]. The parameter settings for the compared algorithms follow the previous works [66, 54, 74].

On synthetic datasets, Prelić's observation (gene) match score and overall match score [66] are adopted to evaluate the bi-clustering algorithms' ability to discover the implanted (true) bi-clusters. Let $\mathcal{C}$ and $\mathcal{C}^*$ denote the set of output bi-clusters from an algorithm and the set of true bi-clusters for a dataset respectively. The observation match score of $\mathcal{C}$ with respect to the target $\mathcal{C}^*$ is defined as $\mathrm{score}_G(\mathcal{C}, \mathcal{C}^*) = \frac{1}{|\mathcal{C}|} \sum_{(G_1,S_1)\in\mathcal{C}} \max_{(G_1^*,S_1^*)\in\mathcal{C}^*} \frac{|G_1 \cap G_1^*|}{|G_1 \cup G_1^*|}$, which is the average of the maximum observation match scores of bi-clusters in $\mathcal{C}$ with respect to the target bi-clusters. The feature match score $\mathrm{score}_S(\mathcal{C}, \mathcal{C}^*)$ can be similarly defined by replacing observation sets with the corresponding feature sets in the above. The overall match score is then defined as in (2.2).

As for the parameter setting of SLLB, we set $\beta_1 = 0.1$, $\beta_2 = 0.05$ on all the noise resistance experiments, and we set $\beta_1 = 0.1$, $\beta_2 = 0.3$ on all the overlapping experiments. Details of the parameter selection methodology are described in the discussion section below.

Noise resistance test: This experiment investigates the different bi-clustering algorithms' ability to recover implanted bi-clusters under different noise levels. Following Prelić's testing strategy, we first generate a 100×50 background matrix, based on a standard normal distribution, and then embed ten 10×5 non-overlapping linear coherent bi-clusters along the diagonal. Then, for each vector of the five expression values, we set the first two to be down-regulated, the last two to be up-regulated, and the middle one to be non-regulated. Lastly, we add noise of six different levels ($\ell = 0.00, 0.05, 0.10, 0.15, 0.20, 0.25$) to the embedded bi-clusters by perturbing the entry values so that the resultant values are ℓ away from their original values. The generation is repeated ten times. Based on the same simulation

process, we generate additive bi-clusters on synthetic datasets when we compare the bi-clustering algorithms with respect to their performance at recovering only additive bi-clusters (which is a special case of linear coherent bi-clusters).

Figure 2.16 shows the observation match scores of the bi-clusters discovered by the eight algorithms at six different noise levels. Figures C.1 and C.2 in the Appendix C demonstrates the overall match scores and observation discovery rates obtained (defined as the percentage of observations in the output bi-clusters over all the observations in the true bi-clusters). From these figures, it is clearly shown that SLLB outperformes all the other seven algorithms; QUBIC, LinCoh and ISA rank the second, third, and forth, and the other three performed quite poorly. Note that by simply outputting more bi-clusters, the observation discovery rate can be trivially lifted, therefore it is only a useful measurement in conjunction with match scores.

Overlapping test: Bi-clusters may overlap in terms of either observations or features. Take gene expression as an example: some genes can participate in multiple biological processes, resulting in bi-clusters that overlap with common genes in an expression matrix. The same is also true with respect to sample overlapping. Therefore, this experiment intends to examine the ability of the bi-clustering algorithms to recover overlapping bi-clusters. We again consider type-(f) linear coherent bi-clusters and type-(d) additive bi-clusters, at a fixed noise level of $\ell = 0.1$. We generate ten 100×50 matrices based on a standard normal distribution. In each matrix, two 10×10 bi-clusters are embedded, with overlapping size: 0×0, 1×1, 2×2, 3×3, 4×4, and 5×5. In the case of gene expression, we assume that these overlapping genes to obey a reasonable logic such as the AND gate and the OR gate which leads to a *union* and an *additive* behavior. So the overlapped entries in the union overlapping area preserve linear coherency in both bi-clusters and in the additive overlap model, these entries are assigned by the sum of the gene expression levels from both bi-clusters. The observation match scores of the eight bi-clustering algorithms in this additive overlapping experiment are shown in Figure 2.17. Figures C.3 and C.4 in Appendix C plot the overall match scores and observation discovery rates under the additive overlap model. The results of the

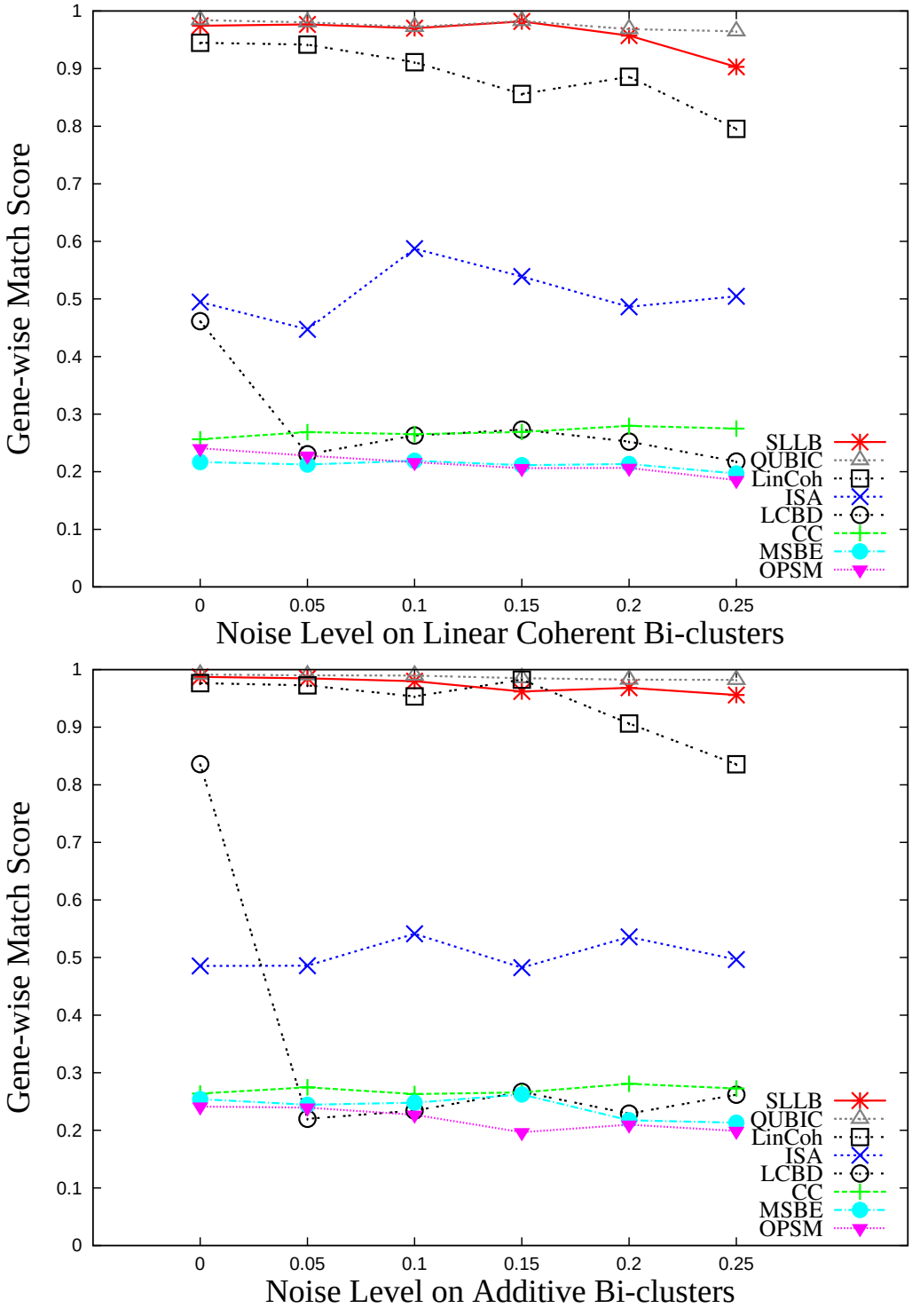


Figure 2.16: The observation match scores of the eight algorithms on recovering linear coherent bi-clusters and additive bi-clusters at six different noise levels.

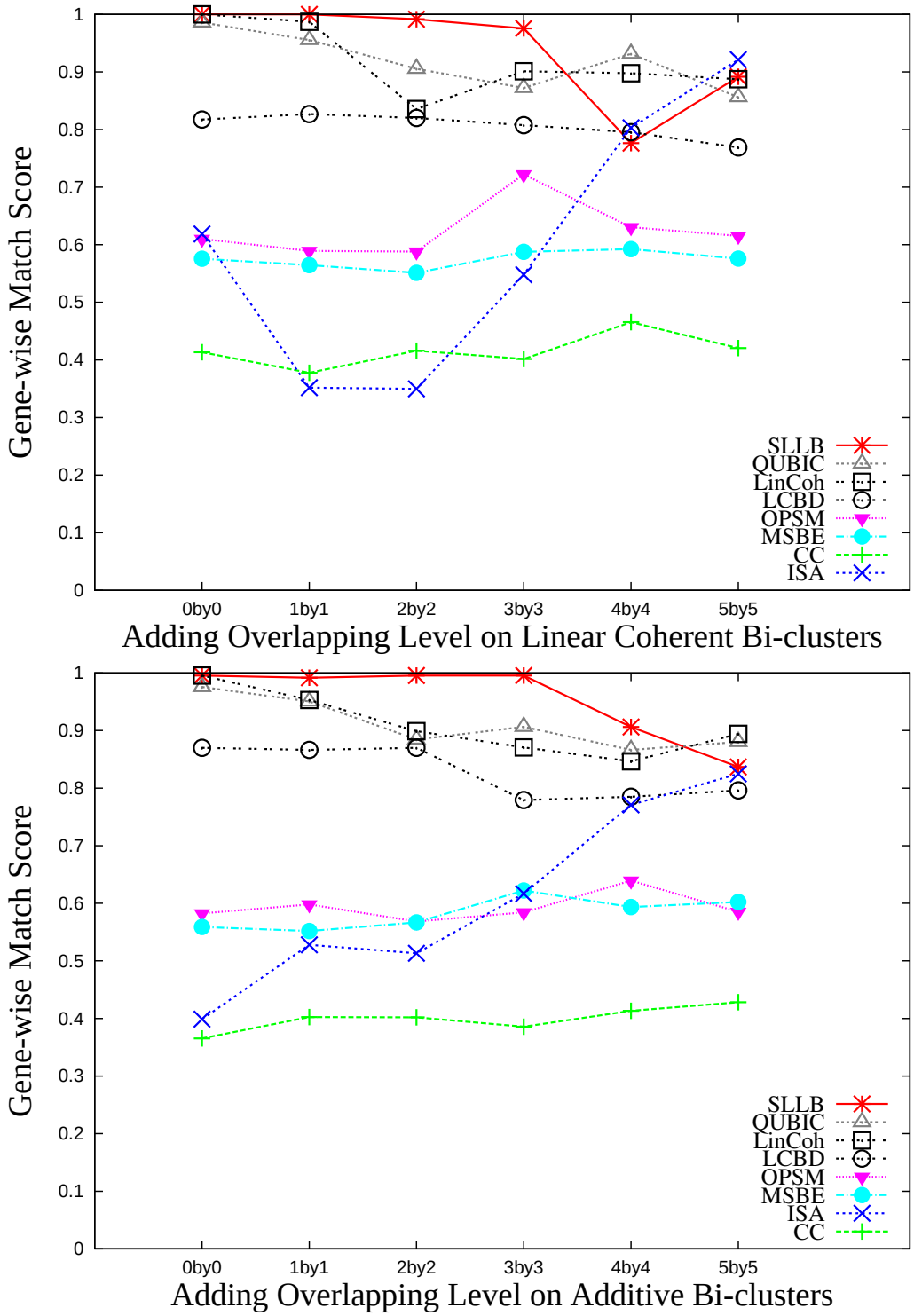


Figure 2.17: The observation match scores of the eight algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the additive overlap model.

additive overlap model are shown in Figures C.5, C.6, and C.7 in the Appendix C. From all these results, we can conclude that SLLB outperforms the other seven algorithms. LinCoh's performance is slightly worse than SLLB; QUBIC, OPSM and MSBE perform worse, but similarly to each other; LCBD and CC performed the worst; and ISA demonstrates varying performance.

2.3.3 Results on Real Data

We then tested all of the algorithms on two real gene expression microarray datasets: *Saccharomyces cerevisiae* (yeast) and *Escherichia coli* (e.coli) respectively. On real datasets, the quality of bi-clusters are evaluated by known biological pathways, defined in the GO functional classification scheme [1], the KEGG pathways [41], the MIPS yeast functional categories [68] (for yeast dataset), and the EcoCyc database [46] (for e.coli dataset), to obtain their *gene functional enrichment score* as implemented in [53]. The average correlation coefficient is also used for evaluating the generated bi-clusters on real datasets.

We obtain the yeast dataset from [23]. It contains 2993 genes on 173 samples; the e.coli dataset is obtained from [21], (version 4 built 3). It contains initially 4217 genes on 264 samples. For the e.coli dataset, after removing genes that do not have any significant expression deviations, we obtain 3016 genes. This pre-processing step ensures that all eight bi-clustering algorithms can run on the dataset. Again, we use the gene functional enrichment score [53] in (2.3) to measure the performance of different algorithms. Such a probability is taken as the P-value of the output bi-cluster enriched with genes from that functional class [53]. The P-value of the output bi-cluster is defined as the smallest P-value over all functional classes. The smaller the P-value of a bi-cluster the more likely its genes come from the same biological process. We calculate for each algorithm the fraction of its output bi-clusters whose P-values are smaller than a significance cutoff α. As for parameter setting of SLLB, we set $\beta_1 = 0.3$, $\beta_2 = 1.5$ for yeast dataset, and $\beta_1 = 0.1$, $\beta_2 = 0.5$ for e.coli dataset. Details of the parameter selection methodology are described in the discussion section below.

In Figure 2.18, the eight algorithms are compared using six different P-value

cutoffs, evaluated on the GO database. Results on the KEGG, MIPS, and Regulon databases are in Figures 2.19 and 2.20. These results indicates that SLLB performs consistently well; QUBIC and LinCoh performs stable but worse than SLLB, OPSM and ISA does not perform consistently on the two datasets across databases; and that LCBD, MSBE and CC does not perform as well as the other three algorithms.

One potential issue with the P-value based performance measurement is that P-values are sensitive to the bi-cluster size [53]; in general, this measurement favor bi-clusters with larger size. For example, in Table C.1, it is shown that OPSM find bi-clusters that contain an extremely large number of genes and very few samples. Bi-clusters of this kind are close to trivial bi-clusters (gene or sample set size close to 0) but with a large number of genes, its enrichment P value can be easily lifted. On the contrary, although our SLLB algorithm generates bi-clusters with a large number of genes, the number of samples it generates is also large, which indicates more confident linear coherence. In the last column of Table C.1, the number of unique functional terms enriched by the produced bi-clusters are listed. When measured by the gene enrichment significance score, OPSM performed very well on yeast dataset (Figure 2.18, left), but its bi-clusters only cover one functional term on the GO and KEGG databases and two terms on MIPS database. This suggests that the bi-clustering result can be biased to a group of correlated genes, missed by the P-value based significance test.

Considering these two potential issues, we can see that the P-value based evaluation is meaningful but has limitations. So we propose to use the average absolute correlation coefficient over all gene pairs in a bi-cluster as an alternative assessment of the quality. However, note that the numbers of samples in the bi-clusters generated by some algorithms are much smaller than others; see Table C.1. Therefore, to compare algorithms in a less sample-size biased way, we replace for each bi-cluster its average absolute correlation coefficient by the 99% confidence threshold using the number of samples in the bi-cluster [71, 74]. These values are plotted in Figure 2.21.

Figure 2.21 shows that SLLB, LinCoh, and OPSM have similarly good perfor-

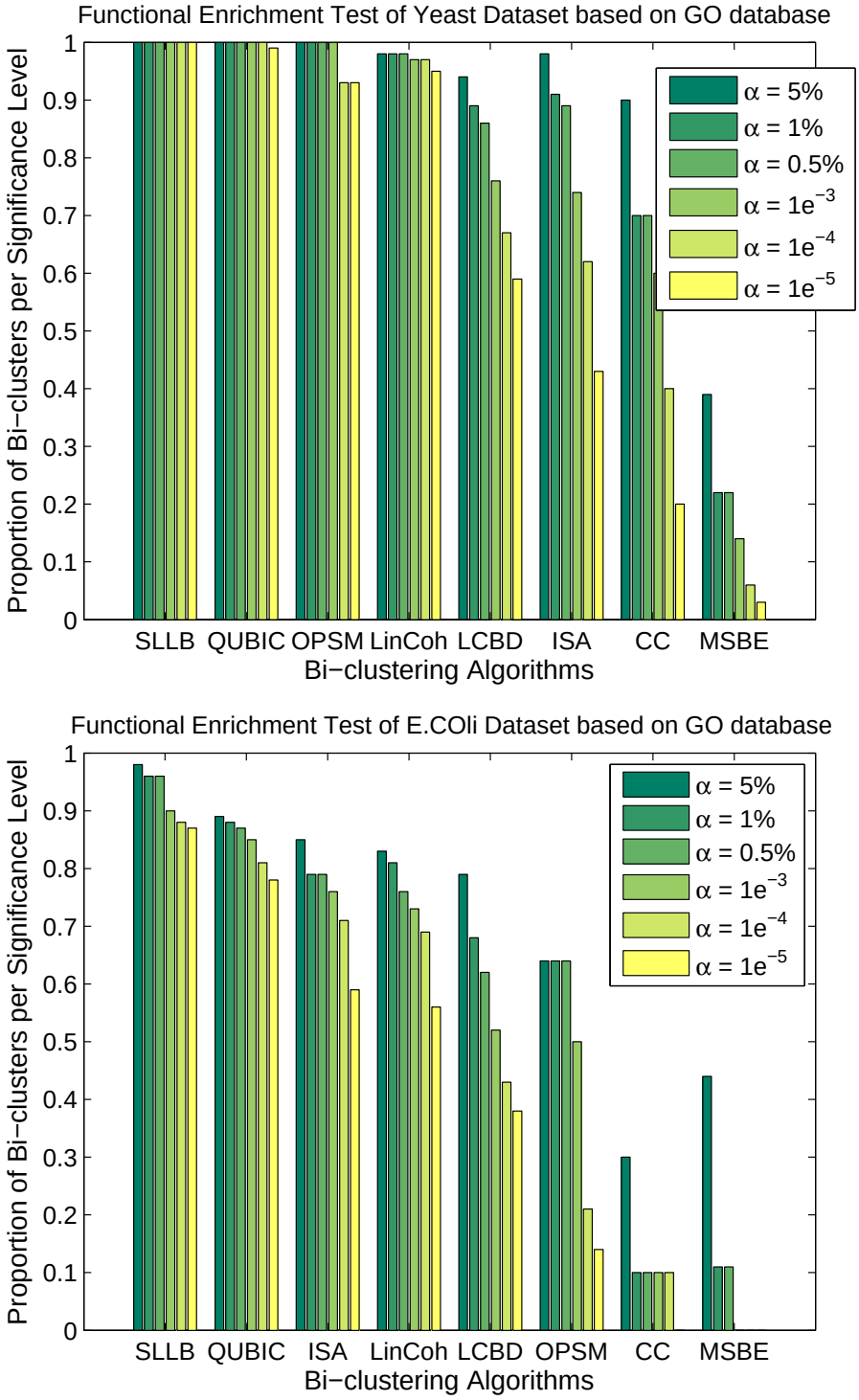


Figure 2.18: Portions of discovered bi-clusters by the eight algorithms on the two real datasets that are significantly enriched in the GO biological process, using six different P-value cutoffs.

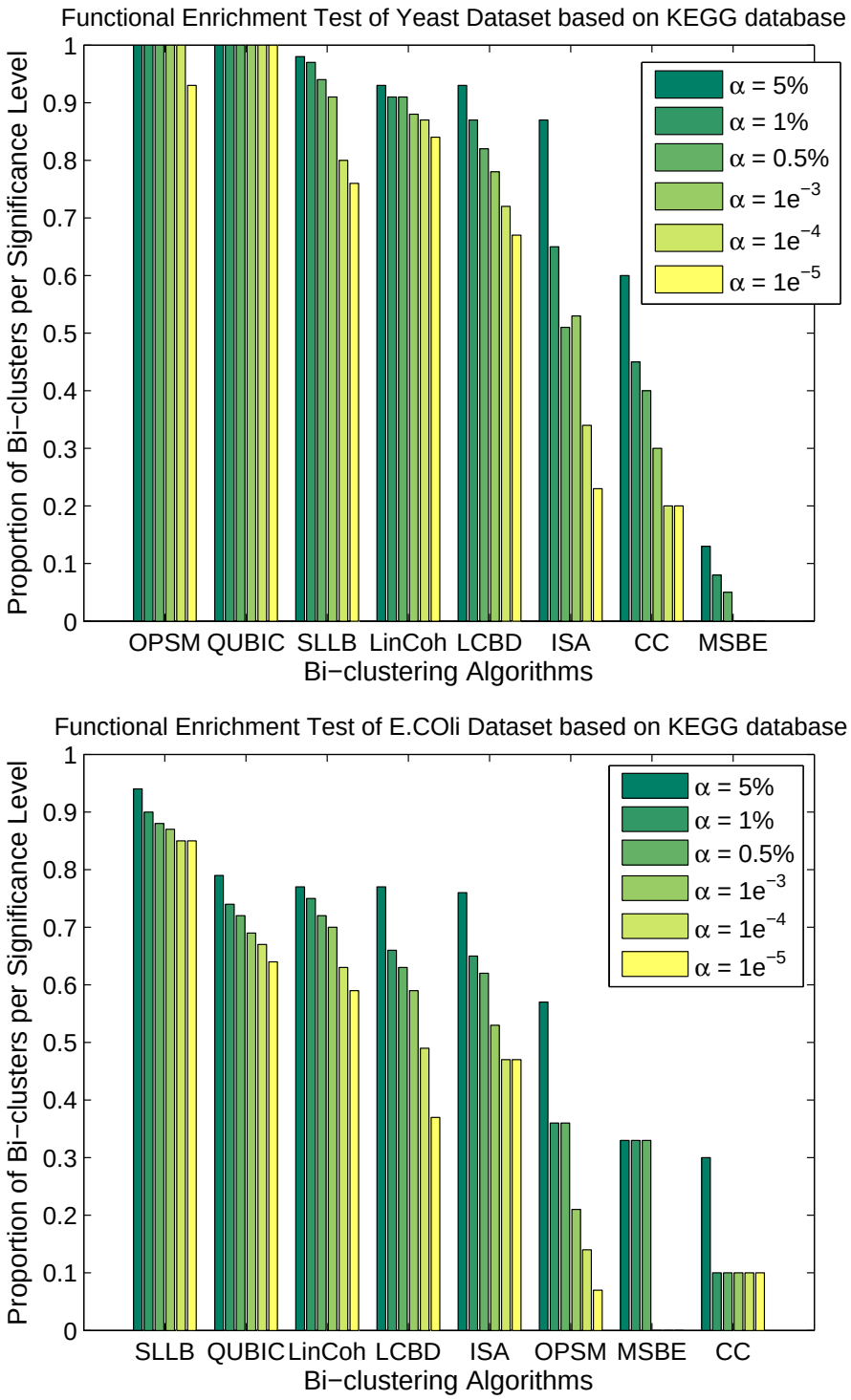


Figure 2.19: Portions of discovered bi-clusters by the eight algorithms on the two real datasets that are significantly enriched in the KEGG pathway, using six different P-value cutoffs.

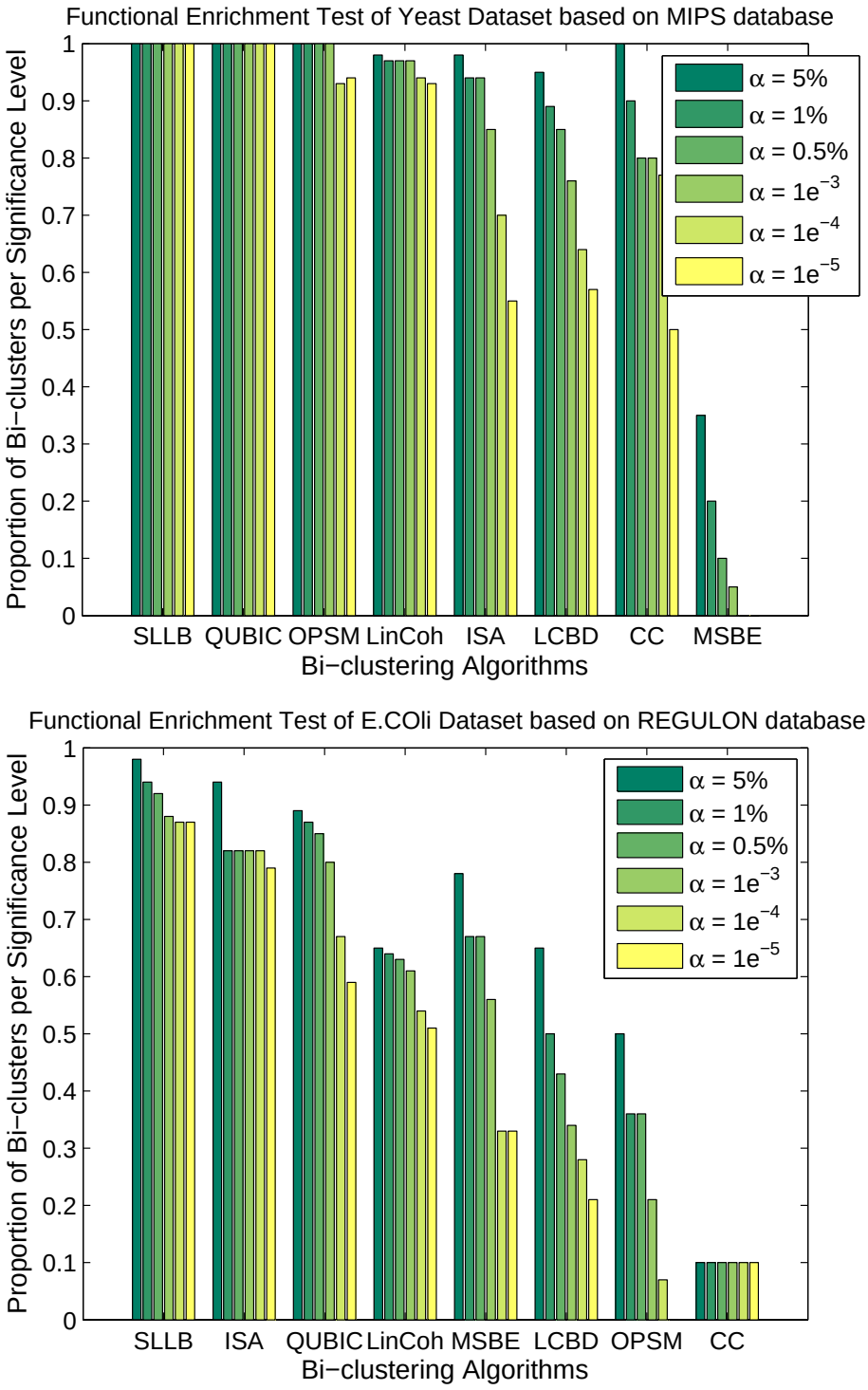


Figure 2.20: Portions of discovered bi-clusters by the eight algorithms on the two real datasets that are significantly enriched in the MIPS pathway experimentally verified REGULONS, respectively, using six different P-value cutoffs.

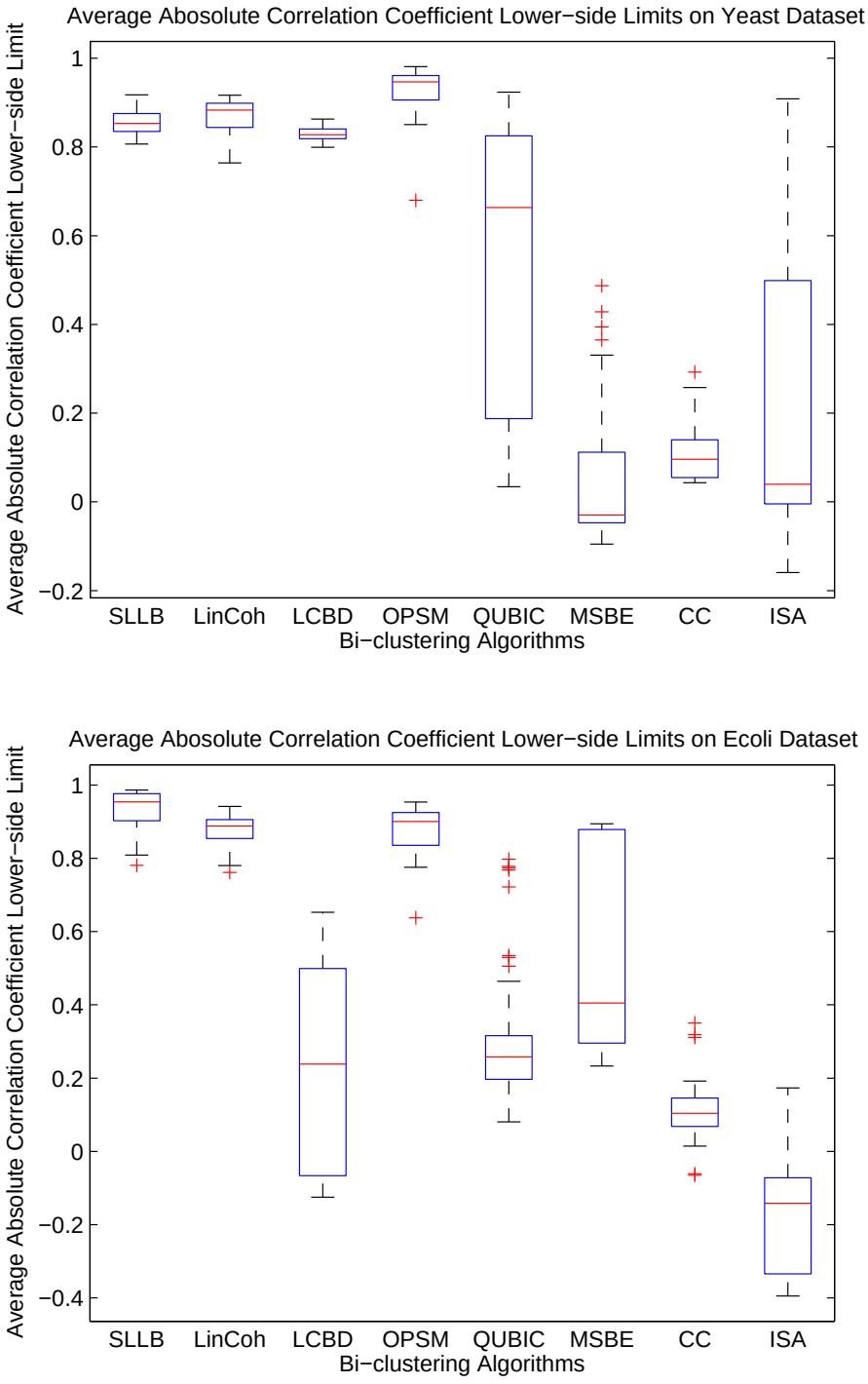


Figure 2.21: Box plots of the average absolute correlation coefficients obtained by the eight bi-clustering algorithms on yeast and e.coli datasets, respectively.

mance while QUBIC, LCBD, MSBE, CC, and ISA performs worse than these three. Note that due to noise effect when profiling genes, itt is hard to reach a very large value of the correlation coefficient.

2.3.4 Discussion

In this section, we proposed a novel bi-clustering algorithm, SLLB, that can discover linear coherent bi-clusters, based on a sparse learning optimization model. The experimental results on both synthetic and real datasets indicate that SLLB is not only able to discover linear coherent bi-clusters effectively, but able to discover meaningful linear coherent bi-clusters that can be verified by biological ground truth. Actually, for many bi-clusters discovered by SLLB, their corresponding gene groups (with size 30-100) generally all belong to the same gene ontology term, which is a striking result. The time complexity of the SLLB algorithm is $O(n^2k)$ where n is the number of observations and k is the number of iterations that SLLB takes to converge, which is very fast compared to algorithms like LinCoh. Note that, while discovering linear coherent bi-clusters, SLLB favors data points corresponding to features that are far from each other in the observation pair 2D space, which is a nice property that can be used for downstream data analysis, for example, in feature clustering, observation-feature relation studies and observation/feature selection.

To set appropriate values for β_1 and β_2, I binary searched $\beta_1 \in [0, 1000]$ and $\beta_2 \in [0, 1000]$ and found the value ranges that produce non-trivial bi-clusters are $\beta_1 \in [0, 0.5]$ and $\beta_2 \in [0, 1.5]$. I then tested different combinations of $\beta_1 = [0.1, 0.5, 1]$ and $\beta_2 = [0.1, 0.5, 1, 1.5]$ and found the results are quite robust to different settings. The final β_1 and β_2 are chosen so that SLLB performs best. When come to practice, considering that β_1 actually controls the size of observation and β_2 controls the size of features in the result bi-clusters, β_1 and β_2 can be determined when prior biological knowledge of bi-cluster size is available.

We suggest that the SLLB algorithm can be used in other machine learning applications such as image clustering, document clustering, and other biology and health care data clustering, as long as observations of a common group have linear

coherence under a subset of features, whereas for different clusters, different feature sets need to be selected. As for future work, we will test SLLB on other applications such as document bi-clustering and image bi-clustering. We will also extend SLLB to be able to consider other relations between observations in addition to linear coherent relations.

Chapter 3

Directed Bio-relation Discovery

The previous bi-clustering algorithms attempted to identify undirected relations between genes, while in other bio-network inference problems, we are interested in identifying directed relations between objects. The bio-network discovery problems I have been investigated, have been in the areas of genetic regulatory networks, drug-target networks, and metabolite networks. In the genetic regulatory network problem, we want to recover the directional regulation relation between transcription factors (TFs) and regulated genes; in the drug-target network problem, we want to recover the directional drug-target binding relations. Here I will adopt and extend some existing methods such as least absolute shrinkage and selection operator (LASSO) and the feature selection support vector machine (SVM) to address a key challenge in bio-network discovery, the sparsity/feature selection problem.

3.1 Gene Regulatory Network Inference

Genes play a central role in controlling the function of cells. Rather than acting individually, genes and their products cooperate with each other in the form of a dynamic gene regulatory network (GRN). GRNs serve as the blueprints of life. Revealing these complex blueprints is critical to understanding life at a system level and helps the development of biological understandings, drug design, health care, etc. One of the key mechanisms of gene regulation takes place at the RNA transcription level. With the emergence and development of high-throughput gene profiling technology and ChIP-on-chip technology, GRN research has been boosted to

a large extent, but reliable GRNs are still being sought in the research community. The goal of such research is to discover the causal control relationships between genes, which would offer a fundamental understanding of how biological processes are coordinated in the cells.

To model gene regulatory networks from expression data, various computational approaches have been proposed in the literature during the past decade. Among these approaches, many use linear models to express dependence between time series profiles. For example, D'Haeseleer et al. [17] proposed a straightforward linear model; De Jong et al. [40] and Chen et al. [12] studied linear differential equations for modeling gene regulatory networks. However, all of these methods suffer from the risk of over-fitting, since the parameters their models fit are proportional to the size of the data itself. Other linear approaches took advantage of sparseness of the regulatory relationships between genes to overcome the risk of over-fitting. These models employ the idea that any one gene is regulated by only a small subset of the other genes. De Hoon et al. [32] proposed in the linear system to use Akaike's Information Criterion (AIC) to determine the nonzero coefficients. Similarly, Li and Yang [52] used L_1 regularization to select features of the linear parent set.

Learning different forms of (dynamic) Bayesian network structure is another popular approach in gene regulatory network structure inference. Bayesian networks are graphical representations of the causal relationships of a set of variables. A Bayesian network provides a well formed probabilistic framework for representing and inferring probabilistic relationships. Dynamic Bayesian networks are extension of standard Bayesian network models to represent time-series. Generally, there are two approaches in learning the structure of a Bayesian network from data. The first approach is a score-based approach where a heuristic search is performed through the space of causal network structures to identify the most likely structure explaining the data. The second approach is a constraint-based approach where conditional independence tests are used to determine whether a direct causal relationship should be postulated between two variables. In the literature, many variants of these techniques have been applied to gene regulatory network inference, including search-based approaches [29, 94, 86], information-theoretic approaches

[13], parameterizing based approaches [70], and conventional dynamic Bayesian network learning approaches [4, 99].

The above approaches have achieved some promising results, but their effectiveness is severely constrained by the fundamental limitation of the amount of data available relative to the large number of parameters estimated (e.g. distinct parameters used to predict the expression level of each gene given other genes). This difficulty is inherent to the task because of lacking background knowledge and biologically relevant assumptions. More expression data of orders of magnitude is required for naive estimation approaches.

Nearly all proposed approaches using either linear modeling or Bayesian network structure learning have one common shortcoming that they attempt to determine the regulation structure for each target gene independently, while it is well known that genes that share the same expression pattern are likely to be involved in the same regulatory process, and therefore share the same (or at least a similar) set of regulators [17]. Although a few investigators, such as van Someren et al. (2000), have previously proposed to group genes with similar expression profiles into a single prototypical ”gene”, and then model the relations between prototypical genes instead of modeling the genes individually. This is a somewhat oversimplified approach that ultimately ignores the individual differences between genes in the same group, and puts a particular high requirement on the clustering step. Guo *et al.* [26] tried to employ biologically significant knowledge about co-regulation to improve the inference of the underlying gene regulatory network from expression data, by proposing a novel approach for predicting the regulators for a given group of genes with similar mRNA expression patterns. They implemented this intention by minimizing a globally shared regularized prediction risk that encourages similar genes to share regulators. The novelty of the approach is to first cluster the genes based on their time series expression profiles, and then minimize a loss determined on a set of global indicator variables associated with the common set of possible regulatory variables. The performance of this approach on both synthetic data and the cell cycle time-series gene expression data of [16] was quite promising that important transcription factors (TFs) in the cell cycle genes were identified more accurately.

In this work, I extend this previous work [26] in four aspects: 1) kernelize the original linear models, 2) use cubic spline interpolation and clustering information to more accurately address the time lag problem, 3) take the within-cluster gene competition effects into account, and 4) use cross entropy loss as an alternative to the $L2$ norm loss in the objective function.

3.1.1 Method

The core of this work is to use the kernel model instead of the linear regression model to implicitly consider the non-linear relations among transcription factors so that more accurate TF-gene causal relations can be discovered. The previous linear model approaches implicitly admit that multiple transcription factors cooperate but independently regulate target genes. This is not true in cases when multiple transcription factors in the form of single protein complexes regulate target genes. I therefore propose to kernelize the linear model to map the independent linear relations between transcription factors to more complex relations in high dimensional spaces. To derive a good kernel model, three other approaches are adopted: first, use cubic spline interpolation and gene clustering information to address the TF-gene time lag problem; two, take the within-cluster gene competition into account; three, use cross entropy loss as an alternative to the $L2$ norm loss in the objective function. This work was published in [73].

The linear model and its kernelization. I first re-introduce the linear models described in the previous work [26]. Consider an $n \times t$ matrix Y of time series gene expression data, where each column corresponds to the expression levels of a single gene measured over a series of n time points, and each row corresponds to a single time point measured over a set of t. For each gene, I want to identify which other genes measured in Y are likely to be it regulators. The fundamental hypothesis here is that the expression levels of a regulator gene should be predictive of the expression levels for a regulated target gene, possibly subject to time lag and the presence of co-regulators or absence of inhibitors. The simplest linear prediction approach assumes for a target expression profile y_j given by an $n \times 1$ column vector from Y , I have a set of candidate regulator profiles stored in an $n \times k$ matrix X_j

consisting of k distinct columns selected from Y. The potential regulators can be determined by solving for the combination weights of the regulator profiles that best reconstruct the target profile

$$\min_{\mathbf{w}_j} \|X_j \mathbf{w}_j - \mathbf{y}_j\|_2^2, \tag{3.1}$$

where the $k \times 1$ vector of combination weights w_j describes how much each of the k regulator genes in X_j contribute to best fit the target expression levels y_j , and the quality of the fit can be assessed by the residual error in (3.1).

Because the set of candidate regulators for a given gene is usually much larger than the number of time points, a large set of combination weights w_j need to be inferred from a limited amount of data. Moreover, only a tiny fraction of the candidate regulators are expected to be true regulators for any given gene, which means most of the weights should be set to 0 to indicate non-regulation. Therefore, I use the $L1$ norm regularization (rather than the traditional $L2$ norm) to perform feature selection, which is well known in machine learning literature [61, 77, 26] for its effectiveness. In this approach, one adds a penalty to the risk (the reconstruction objective) which encourages small values for w_j :

$$\min_{\mathbf{w}_j} \|X \mathbf{w}_j - \mathbf{y}_j\|_2^2 + \alpha \|\mathbf{w}_j\|_1, \tag{3.2}$$

where α is a parameter that trades off the influence of the risk with the regularizer. The regularizer encourages many of the weights to become exactly zero in the solution.

Considering that genes with similar expression patterns are likely co-regulated and involved in the same functional process, I proposed to first cluster the target genes based on their expression profiles. (A straightforward K-means method is implemented in our case.) Then, for each cluster, I want to identify a set of regulators that is shared among the entire set of genes in the cluster, while still allowing for differences among the regulation of individual genes. Similar to [25], I introduced a set of auxiliary indicator variables to control global feature selection, and used a global regularization scheme on auxiliary selection variables to help identify the

common candidate regulators among a group of target genes with similar expression profiles. Given that there is much more data available for sets of similar genes, as opposed to individual genes, I hope that the common regulators can be more accurately identified. Specifically, for a set of target genes $Y = \mathbf{y}_1, ..., \mathbf{y}_m$, I want to identify a common set of regulators from the set of candidates $X = \mathbf{x}_1, ..., \mathbf{x}_l$. Define a set of indicator variables $\eta = \eta_1, ..., \eta_l{}^T$, corresponding to the candidate set $X = \mathbf{x}_1, ..., \mathbf{x}_l$, such that each $\eta_i \in \{0, 1\}$ indicates whether a regulator X_i is selected as an active regulator. Let $N = diag(\eta)$. Then, I can form a globally regularized version of the minimization problem by introducing the selection variables η and adding a new global regularization term on these variables:

$$\min_{\eta \in \{0,1\}^n} \min_{\mathbf{w}_j} \sum_j (\|XN\mathbf{w}_j - \tilde{\mathbf{y}}_j\|_2^2 + \alpha\|\mathbf{w}_j\|_1) + \lambda \mathbf{u}^T \eta \tag{3.3}$$

where $\mathbf{u}$ is a positive weight vector that allows one to incorporate prior knowledge about the importance of each global feature (which is simply set to 1's in our experiments). The global regularization term $\lambda \mathbf{u}^T \eta$ is an $L0$ norm regularizer that automatically forces a sparse solution that selects only a small set of global features for the set of target genes in a cluster. The local $L1$ norm regularizer $\alpha\|\mathbf{w}_j\|$, however, will still make individual choices of regulators for each target gene; choosing these regulators from the globally selected features identified by η.

The linear models in (3.1) - (3.3) assume that if a target gene (or a set of target genes) has multiple regulators, each regulator can independently regulate the target genes and their overall regulation effect is a weighted addition of their individual effects. This is not true in cases where different transcription factors need to form a single protein complex and possibly cooperate with other such protein complexes to regulate the target genes. Interpreted in another way, this means that the linear models only consider the so called *OR* gates regulation, while in real cases, the regulation rules are likely obey the so-called canalyzing rules [43, 62, 65, 44], which are combinations of *OR* gates and *AND* gates. It immediately follows that one needs to consider more complex relations between transcription factors than the existing linear models. However, to enumerate all the canalyzing rules over *OR* gates and *AND* gates is computationally impractical, therefore, I propose to kernelize the lin-

ear model to map the independent linear relations between transcription factors to more complex relations in high dimensional space. I hope the kernelized model can avoid the problem of enumerating all the canalyzing regulations and can discover the non-linear relations between transcription factors to better construct the target gene profile and more accurately infer the TF-gene causal relations.

To kernelize the $L2$ norm term in (3.3), let $\mathbf{w}_j = NX^T\mathbf{b}_j$. To kernelized the $L1$ norm term, I consider the trick proposed in [57] as follows:

$$\|\mathbf{w}_j\|_1 = \sum_{p=1}^{k} |w_p| = \min_{\gamma \geq 0} \frac{1}{2} \sum_{p=1}^{k} \frac{w_p^2}{\gamma_p} + \gamma_p \tag{3.4}$$

Substitution $\mathbf{w}_j$ into (3.4), we get:

$$\begin{aligned} \min_{\eta \in \{0,1\}^n} \min_{\mathbf{b}_j} \min_{\gamma_j \geq 0} & \sum_j \|XNN^TX^T\mathbf{b}_j - \tilde{\mathbf{y}}_j\|_2^2 \\ & + \alpha \sum_j (\frac{1}{2}\mathbf{b}_j^T XN \triangle (\gamma_j)^{-1} N^T X^T \mathbf{b}_j + \gamma_j^T \mathbf{1}) \\ & + \mathbf{u}^T \eta \end{aligned} \tag{3.5}$$

Let $\Psi = XN$ and $\tilde{\Psi} = XN \triangle (\gamma_j)^{-1/2}$, (3.5) can be written as:

$$\begin{aligned} \min_{\eta \in \{0,1\}^n} \min_{\mathbf{b}_j} \min_{\gamma_j \geq 0} & \sum_j \|\Psi\Psi^T\mathbf{b}_j - \tilde{\mathbf{y}}_j\|_2^2 \\ & + \alpha \sum_j (\frac{1}{2}\mathbf{b}_j^T \tilde{\Psi}\tilde{\Psi}^T \mathbf{b}_j + \gamma_j^T \mathbf{1}) + \mathbf{u}^T \eta \end{aligned} \tag{3.6}$$

Let $K = \Psi\Psi^T$ and $\tilde{K} = \tilde{\Psi}\tilde{\Psi}^T$, so that each element K_{ij} in matrix K is the inner product of the i'th and j'th row of Ψ, namely $K_{ij} = \Psi_{i:}\Psi_{j:}^T$. Similarly, $\tilde{K}_{ij} = \tilde{\Psi}_{i:}\tilde{\Psi}_{j:}^T$. Because K and $\tilde{K}$ are both symmetric and positive semi-definite, according to the *Mercer*'s theorem, K and $\tilde{K}$ can be expressed as dot products in a high dimensional space. In our case, I use the widely adopted ***RBF*** kernel in which$K_{ij} = e^{\frac{-\|\Psi_{i:} - \Psi_{j:}\|^2}{2\sigma^2}}$, and similarly, $\tilde{K}_{ij} = e^{\frac{-\|\tilde{\Psi}_{i:} - \tilde{\Psi}_{j:}\|^2}{2\sigma^2}}$. Substituting the two ***RBF*** kernel matrices into model (3.6), we get its kernelized version:

$$\begin{aligned} \min_{\eta \in \{0,1\}^n} \min_{\mathbf{b}_j} \min_{\gamma_j \geq 0} & \sum_j \|K\mathbf{b}_j - \tilde{\mathbf{y}}_j\|_2^2 \\ & + \alpha \sum_j (\frac{1}{2}\mathbf{b}_j^T \tilde{K}\mathbf{b}_j + \gamma_j^T \mathbf{1}) + \mathbf{u}^T \eta \end{aligned} \tag{3.7}$$

The feature selection parameter now becomes γ_j (not $\mathbf{w}_j$), where $\gamma_j = 0$ indicates the corresponding regulator is not selected by the target gene. Equation (3.7) encodes a min-min-min integer optimization problem. Unfortunately, integer optimization problems of this form are generally NP-hard. To attempt to solve the problem efficiently, I first relax it into an optimization over continuous variables, by relaxing each $\eta_i \in \{0, 1\}$ to be continuous $\eta_i \in [0, 1]$. This leads to solve the following relaxed min-min-min optimization:

$$\min_{\eta} \min_{\mathbf{b}_j} \min_{\gamma_j \geq 0} \sum_j \|K\mathbf{b}_j - \tilde{\mathbf{y}}_j\|_2^2 + \alpha \sum_j (\frac{1}{2}\mathbf{b}_j^T \tilde{K} \mathbf{b}_j + \gamma_j^T \mathbf{1}) + \mathbf{u}^T \eta \tag{3.8}$$

$$s.t. \quad 0 \leq \eta \leq 1.$$

This formulation has actually relaxed the original $L0$ norm regularizer over η into a $L1$ norm regularizer. In this way we maintain feature selection ability, while gaining computational efficiency.

In our implementation, I use two alternating steps: $\min_w$ and $\min_\eta$ for solving (3.3). Each $\min_w$ step is simply a minimization of least squares regression error with $L1$ norm regularization, which can be implemented as a quadratic program [8], or by using a fast grafting algorithm [77]. For the $\min_\eta$ step, I use a quasi-Newton BFGS method to perform the optimization [6]. For (3.8), because it is not obvious how to calculate the gradients of η_j and $\mathbf{b}_j$, I simply use the Matlab optimization toolbox *fmincon* to solve the problem.

Addressing time lags. Notice that neither of the above models account for any time lag between the expression of a regulating gene and the expression of its downstream target. In fact, they implicitly assumes that regulation occurs instantaneously, and would performs poorly at identifying any regulatory relationship that exhibits delayed effects. To cope with this shortcoming, I previously proposed a simple time-shifting method, in which, for each candidate regulator measured in X_j , given by an $n \times 1$ vector x_{ij} , I first computed an optimal shift back in time that best aligns x_{ij} individually with the target y_j.

$$s_{ij}^* = \arg \min_{s \in \{1,2,3\}} \|\mathbf{x}_{ij}(1, ..., n-s) - \mathbf{y}_j(s+1, ..., n)\|_2^2 \tag{3.9}$$

Repeating this for each candidate regulator profile in X_j yields a series of optimal time lags. I then reformulated the expression matrix X_j for the candidate regulators by applying the optimal shift to each column, and truncating the columns to a common length based on the maximum shift, obtaining an $(n - s_{max} \times k)$ time-lag aligned matrix Φ_j . The target expression profile $\mathbf{y}_j$ was then also truncated to a corresponding $(n - s_{max} \times 1)$ vector $\tilde{\mathbf{y}}_j$, where $\tilde{\mathbf{y}}_j = \mathbf{y}_j(s_{max}, ..., n)$.

The above approach has two major problems. First, it is unlikely that the practical time lag is exactly an integer number of time period, but happen anytime within a time period. Second, because each transcription factor may regulate multiple target genes, it is more correct to consider the time lag between a TF and its whole set of target genes rather than a single gene. The time lag between a TF and its whole set of target genes can then be used as the time lag between the TF and each single genes in the target set. Considering there is always noise in microarray data, the TF-Gene set time lag is more meaningful. In practice, I first cluster the gene in different groups, then use cubic spline interpolation to represent all the gene profiles in continuous cubic polynomial functions. The optimal time lag between a TF $\mathbf{x}_{ij}$ and a target gene $\mathbf{y}_j$ is then:

$$s_{ij}^{*} = \arg\min_{s} \int_{s+1}^{n-s} \sum_{p \in j' scluster} |f(\mathbf{x}_{ij}) - f(\mathbf{y}_{jp})| \tag{3.10}$$

where s can be arbitrarily small time step (I use $1/10$ of the original time period). I then use the same procedure to construct Φ_j for each target gene $\mathbf{y}_j$ with time point dense n inverse proportional to s.

Gene competition. To the best of our knowledge, when investigating the TF-gene regulations, almost all the previous researches focus on the TF-gene relations only, while the potential competition effects among genes that share the common set of TFs are neglected. For example, if gene y_1 and gene y_2 are co-activated by a common set of transcription factors X_1, the expression profile of gene y_1 may not only depends on the expression profiles of X_1, rather, the expression profile of y_2 may play a negative role in constructing the expression profile of y_1 in this case. I therefore take this gene competition effect into account in our linear model (3.3) since it is no obvious how to adopt this idea in the kernelized model.

Specifically, consider (3.3), for a cluster of k genes, its corresponding weights W is a $l \times k$ matrix, where each column weight vector indicates how much one of the k genes select the l candidate regulators, and each row weight vector indicates how much a regulator is selected by the genes. To consider the gene competition effect, I can force the sum of each row vector of W to be smaller than a given threshold. To determine the exact values of these thresholds requires prior knowledge or some learning algorithms in the future, I simple set them to be all 1's in our implementation for a primary test.

Cross entropy loss. So far, the above linear model and kernel model all use $L2$ norm loss. In our implementation, I also tried the cross entropy loss in our kernel model, as it is commonly used as an alternative to the $L2$ norm. For an observed vector $\mathbf{y}$ and an estimated vector $\hat{\mathbf{y}}$, the cross entropy loss is defined as:

$$loss_{CrsEtp}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \ln \hat{y}'_i - (1 - y_i) \ln (1 - \hat{y}'_i)$$

where $\hat{y}'_i = \frac{1}{1+e^{-\hat{y}_i}}$.

3.1.2 Experimental Results

I conducted experiments on real cell cycle data to evaluate our approach. In particular, I compared our kernel method (both L_2 loss and cross entropy loss) to global regularization approach (with and without gene competition effect), the standard independent local predication approach, and a prototype based linear regression method adapted from [80]. I applied the methods to inferring the structure of the regulatory network of the yeast cell cycle. In our experiments, I assumed all transcription regulations work through activators, instead of inhibitors; that is, I assumed the w parameters are nonnegative in the linear regressions.

Yeast contains more than 6000 genes, while only a subset of these genes are cell cycle regulated. It is known there are 9 important transcription factors (TFs) that regulate the cell cycle process [76], namely: SWI4, SWI6, MPB1, FKH1, FKH2, NDD1, MCM1, ACE2 and SWI5. Because many gene regulatory relationships in Yeast have already been identified, it is commonly used to evaluate learning approaches that attempt to infer gene regulatory networks from data. Here I use Cho *et*

al.'s data [16], and focus on the task of identifying the subset of regulators from the 9 candidate TFs, for each yeast gene that is cell cycle regulated. I choose a subset of 267 cell cycle regulated genes from the Cho et al. data [16] to clearly evaluate our approach, while I could obtain confirmed regulatory relationships from the previous literature [76, 37], or could obtain potential regulation relationships from existing binding data [76] for 127 genes among them. I re-scaled the expression data to values between 0 and 1, and then clustered the genes into 15 clusters using K-means. (In the images shown in Figure 3.1 and 3.2, the genes are grouped vertically into the clusters. The number of clusters is chosen by using visual judgment to achieve a smooth clustering effect.) Finally, I tested our algorithms on each cluster. After obtaining the w parameters from each algorithm, all the parents indicated by $w > 10^{-5}$ are determined as predicted regulators for the corresponding genes. For a fair comparison, the regularization parameters (α and λ) were chosen to yield the highest F-measure values in each case. Since the regulatory mechanisms are still not known for a portion of the 267 genes, I therefore can only evaluate the results over the 127 genes for which regulatory relationships are presumed known.

Figure 3.1 and Figure 3.2 show the prediction results on 127 genes for all the six algorithms with and without applying the cubic spline interpolation for the time lag problem respectively: locally regularized prediction, prototype based prediction, globally regularized prediction without considering the within-cluster gene competition effects, globally regularized prediction considering the within-cluster gene competition effects, the kernel model prediction, and the kernel model prediction with cross entropy loss. The images compare the performance of the six methods on inferring regulators from among the 9 candidate TFs, and shows how they related to the known TF-based regulatory relationships.

Table 3.1: Results after applying cubic spline interpolation.

Performance comparison	Local	Prototype	Global (no compete)	Global (compete)	Kernel	Kernel (CE loss)
Accuracy(%)	59.9	55.7	68.3	**70.6**	66.9	54.1
Precision(%)	22.1	20.6	26.9	28.3	**31.4**	25.4
Recall(%)	42.5	45.2	37.1	33.9	60.2	**71.0**
F-measure	29.1	28.3	31.2	30.9	**41.3**	37.4

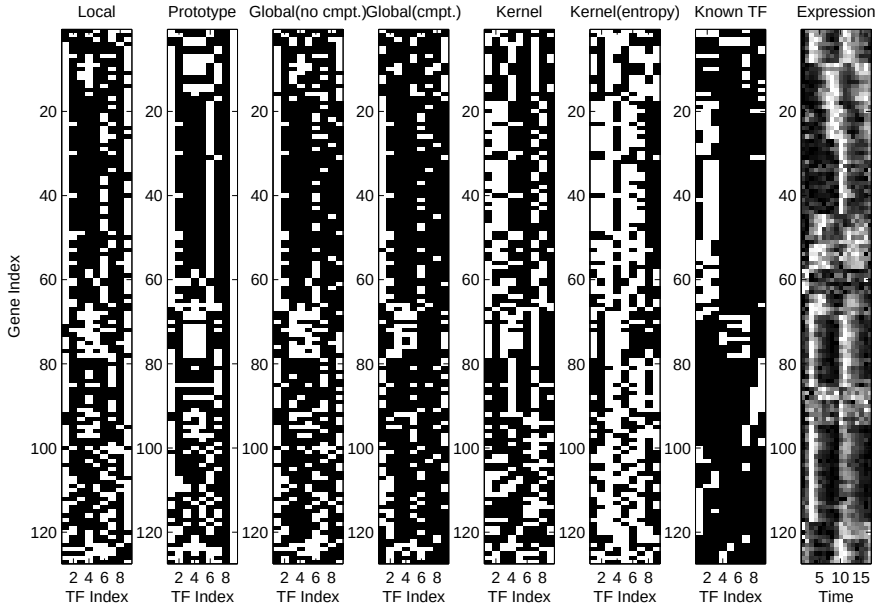


Figure 3.1: Results after applying the cubic spline interpolation for regulation time lag problem on the subset of the real gene expression data from [16], restricted to genes where TF-based regulation information is known or can be inferred from other sources [76, 37]. Rows denote target genes in the synthetic experiment. Columns denote candidate regulators (transcription factors). A white cell denotes a large weight ($w_{ij} > 10^{-5}$) connecting a TF j to a target gene i in the estimated linear model, indicating that j is inferred to regulate i. A black cell denotes a small weight ($w_{ij} \leq 10^{-5}$), indicating that j is not inferred to regulate i. Column 1: local prediction output. Column 2: prototype prediction output. Column 3: global prediction without gene competition output. Column 4: global prediction with gene competition output. Column 5: Kernel method prediction. Column 6: Kernel method prediction with cross entropy loss. Column 6: ground truth regulatory relationships. Column 7: expression level data used as input.

Table 3.2: Results without applying cubic spline interpolation.

Performance comparison	Local	Prototype	Global (no compete)	Global (compete)	Kernel	Kernel (CE loss)
Accuracy(%)	57.5	55.0	**67.7**	67.6	49.7	54.6
Precision(%)	22.1	20.9	**29.7**	29.2	25.1	22.2
Recall(%)	47.5	47.5	48.9	47.5	**81.0**	53.8
F-measure	30.2	29.0	36.9	36.2	**38.4**	31.4

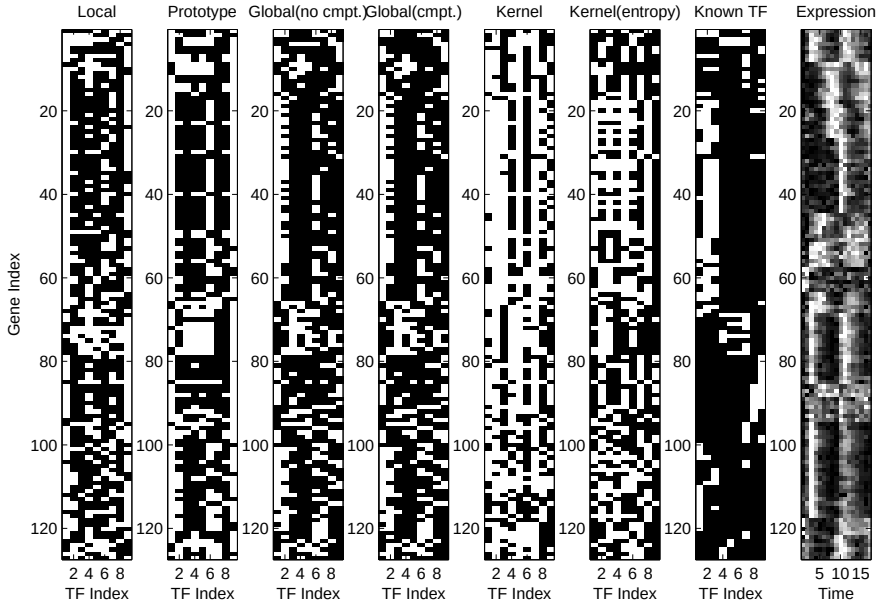


Figure 3.2: Results without applying the cubic spline interpolation for regulation time lag problem on the subset of the real gene expression data from [16], restricted to genes where TF-based regulation information is known or can be inferred from other sources [76, 37]. Rows denote target genes in the synthetic experiment. Columns denote candidate regulators (transcription factors). A white cell denotes a large weight ($w_{ij} > 10^{-5}$) connecting a TF j to a target gene i in the estimated linear model, indicating that j is inferred to regulate i. A black cell denotes a small weight ($w_{ij} \leq 10^{-5}$), indicating that j is not inferred to regulate i. Column 1: local prediction output. Column 2: prototype prediction output. Column 3: global prediction without gene competition output. Column 4: global prediction with gene competition output. Column 5: Kernel method prediction. Column 6: Kernel method prediction with cross entropy loss. Column 6: ground truth regulatory relationships. Column 7: expression level data used as input.

Table 3.1 and Table 3.2 compare the performance of the six algorithms with and without applying the cubic spline interpolation for the time lag problem. The precision score measures true positive predictions (tp) divided by true positives plus false positive predictions (fp). That is, $precision = tp/(tp + fp)$. Similarly, recall score is measured in terms of the number of false negative predictions (fn), and is given by $recall = tp/(tp + fn)$. F-measure is a standard combination of both precision (p) and recall (r), given by $F - measure = 2pr/(p + r)$. The accuracy score measures the proportion of the correct predictions. That is, $accuracy = (tp + tn)/(tp + tn + fp + fn)$.

These results show that the kernel approaches improve the quality of the regulation relation inference in general. The globally regularized approaches has the ability to share regulatory information between genes within a cluster, leading to better noise robustness than the local approach. The kernel approaches outperform the globally regularized approaches in terms of precision, recall and F-measure in Table 3.1. For example, in Figure 1, in the group of genes indexed between 20-50, one can see that a large set of TFs that were not picked out by any other approaches are picked out by the kernel approach (Column 5). Considering the effect of within-cluster gene competition does not lead to significant improvement in our case. But with appropriate upper bound threshold setting and applied in the kernel methods, I believe it would contribute to better prediction performance. The idea of using cubic spline interpolation to better address the time lag problem works pretty well as the overall performance of all the methods are improved.

Although some local errors remain in this region (and elsewhere), clearly the overall quality of the parent prediction has been improved substantially in the kernel method. Overall, the prediction quality achieved by these methods on this data is still somewhat limited, but has improved significantly over the past few years, and in some sense is remarkable given the noise exhibited in the expression profiles.

3.1.3 Discussion

I have proposed a new kernelized version of globally regularized risk minimization objective for learning regulatory networks from gene expression data. Ex-

ploiting the assumption that genes with similar expression patterns are likely to be co-regulated, our approach first clusters the genes, then learns the regulatory relationships by encouraging genes with similar expression patterns to share regulators. Considering in real cases, the TF-gene regulation rules likely obey the so-called canalyzing rules [43, 62, 65, 44], I proposed to kernelize the linear model to map the independent linear relations between transcription factors to more complex relations in high dimensional space. I conjecture that the kernelized model can avoid the problem of enumerating all the canalyzing regulations and can discover the non-linear relations between transcription factors to better construct the target gene profile and more accurately infer the TF-gene causal relations. To address the regulation time lag problem, I proposed to use cubic spline interpolation to represent the discrete gene profiles to continuous profiles and look for the potential time lag between transcription factor and a set of similar genes instead of a single gene. This makes the time lag searching more robust to noise. I also considered the within-cluster gene competition effect that is neglected by most GRN inference methods in the literature.

Our experimental results on yeast cell cycle data show that the kernel approach is more effective at identifying important (transcription factor based) regulatory mechanisms than the standard independent approach, the prototype based approach, and the globally regularized approach. Thus far, I have only considered using gene expression data in the learning process. Further prediction improvements are likely to come from incorporating further sources of biologically relevant data, such as the within-cluster gene competition mRNA upper bound, binding information [76], or other forms of prior knowledge beyond the co-regulation assumption made here. Moreover, as an effective strategy, the kernel method combined with the $L1$ norm feature selection might be extended to resolve other similar problems in bioinformatics area and other research areas.

3.2 Drug-target Network Inference

Proteins operate in highly interconnected networks ("interactome networks") that play a central role in governing cell function. If a protein's conformation is changed, its function can be altered—affecting cell function. Chemical drugs are small molecules that bind to target proteins and change protein conformation, which can ultimately achieve treatment effects. That is, the function of many classes of pharmaceutically useful protein targets, such as enzymes, ion channels, G protein coupled receptors (GPCRs), and nuclear receptors, can be modulated by ligand interaction. Identifying interaction between compounds (drugs, ligands, molecules) and proteins (targets) is therefore a key aspect of genomic drug discovery. Various high-throughput technologies for analyzing the genome, the transcriptome, and the proteome have enhanced our understanding of the spaces populated by protein classes. Meanwhile, the development of high-throughput screening technology has enabled broader exploration of the space of chemical compounds [18, 42, 81]. The goal of the chemical genomics research is to identify potentially useful compounds, such as imaging probes and drug leads, by relating the chemical to the genomic spaces. Unfortunately, our understanding of the relationship between the chemical and genomic spaces remains insufficient. Consider, for example, the PubChem database at NCBI [88], which records information on millions of chemical compounds, yet the number of compounds with known target proteins is limited. The lack of documented protein-chemical interactions suggests that many remain to be discovered, which motivates the need for improved methods for inferring potential drug-target interactions automatically and efficiently.

By elucidating the interaction between proteins and drug molecules, 3D-structure based "docking analysis" has been the principle method used in drug discovery [39, 59, 75]. In this approach, drug-protein binding affinities are modeled by non-covalent intermolecular interactions, such as hydrogen bonding, electrostatic interactions, hydrophobic and Van der Waals forces. By establishing equations that model the physical interaction between a receptor and potential ligand, the potential energy of binding can be calculated. Interaction predictions based on docking anal-

ysis are relatively reliable, hence many docking software tools have been developed, such as DOCK [75], GOLD [39], and AutoDock [59]. Nevertheless, experimental determination of the compound-protein interactions remains challenging [28, 50]. A major disadvantage of these methods is that they require complete 3D structural information for the target, which is usually unavailable in practice, making docking analyses infeasible for genome wide application. For example, only one mammalian member, bovine rhodopsin [64], is known among the GPCRs (G-protein coupled receptors), the modulation of which underlies the action of 30% of the best known commercial drugs [47].

Given the difficulty of experimental determination of compound-protein interaction, there is significant motivation to develop effective *in silico* prediction methods that can both provide new predictions for experimental verification as well as supporting evidence for experimental results. To predict compound-protein interactions, various computational approaches have been developed. Although docking based prediction [14, 67] is most widely used, as in full docking analysis above, it can only be applied when the complete 3D structure of a protein is already known, hence it cannot be deployed widely. Keiser et al. [45] propose to use the known structure of a set of ligands to predict target protein families. However, such predictions do not take advantage of available protein sequence information, and are thus limited to those between known ligands and different protein families. Campillos et al. [10] propose to predict drug-target interaction based on similarities to side-effects of known drugs. Some predictions of this approach have been verified by *in vitro* binding assays, but the approach remains limited to predictions involving drugs with known side-effects. Yamanishi et al. [91] have investigated the relationship between drug chemical structure, target protein sequence, and drug-target network topology, and developed a regression-based learning method for predicting unknown drug-target interactions. In particular, they integrate chemical and genomic spaces in a unified space (referred to as the "pharmacological space") wherein chemical-chemical, protein-protein, and chemical-protein similarities can be modeled.

Most recently, classification methods have been widely adopted in drug-target

prediction [60, 38, 7]. These methods are based on first inducing similarities between targets and/or drugs, then using these similarities to provide kernel matrices for classification methods, such as support vector machines (SVMs), which can provide predictions for novel drug-target interactions. Such classification methods have been developed from both the drug and the target perspective. For drug-side prediction, drug-drug similarities are first obtained, based on structural or pharmacological information and a known bipartite drug-target interaction graph. This information is then used to predict whether a new drug with known structural or pharmacological information interacts with one of the known target proteins. For target-side prediction, target-target similarities are first obtained, based for example on using the normalized Smith-Waterman score [78] to measure the similarity of two amino acid sequences [91, 7, 92]. This information is then used to predict whether a new target with known similarity to existing targets interacts with one of the known drugs. In this work, we focus on target-side prediction, since this approach can be applied to predicting drug interaction for newly discovered proteins—e.g., proteins for which only sequence information is available. However, when generalizing drug-target predictions from known to newly discovered proteins, we note that overall sequence similarities between proteins masks important interaction information: since drugs are usually much smaller molecules than proteins, drug-target binding sites must comprise only small local regions of the protein. With this in mind, instead of using an overall sequence similarity between proteins, we first attempt to identify key local binding regions from common subsequences of proteins that interact with the same drug. These key subsequences are then used to provide a vectorized feature representation for a protein sequence that can be used in the training and testing phases of a classification method (below we focus on support vector machine classifiers). We believe that using key subsequences (i.e. potential binding regions) as features for the target proteins is a more direct and meaningful representation for drug interaction prediction than previous methods. An additional advantage of using an explicit vector representation of targets, as opposed to assessing similarity based on Smith-Waterman score as above, is that these vectors can be implicitly mapped into higher dimensional spaces, thus

increasing the effectiveness of a kernel-based prediction method. This work was published in [90].

3.2.1 Method

Target Vectorization Our framework for solving the classification problem is similar to Bleakley et al.'s approach [7]. There, the similarity between a pair of target sequence is calculated based on the normalized Smith-Waterman algorithm [78]:

$$SW_{norm}(t,t') = \frac{SW(t,t')}{\sqrt{SW(t,t)}\sqrt{SW(t',t')}}, \tag{3.11}$$

where t and t' are two sequences and $SW(\cdot,\cdot)$ denotes the original Smith-Waterman score [7]. Since the normalized Smith-Waterman algorithm is based on a local sequence alignment and only calculates the similarity based on the most similar sequence piece, such an approach might not include the key sequence regions that interact with the drug compound. (Recall that drugs are small compounds and generally only interact with a few amino acids.) Our approach overcomes this drawback by considering all (long) common substrings between a sequence pair, and computes their similarity based on these *feature substrings*.

In detail, suppose we are given a new target sequence and a training data set consisting of m drugs and n targets, with the corresponding drug-target interaction information, drug 3D structures, and amino acid sequences. The aim is to predict which of the existing drugs in the training set interacts with a new target t_c. Similar to [7], our approach considers one drug at a time. For a drug d_i, we know the set of targets $(t_{i1}, t_{i2}, ..., t_{in_i})$ it interacts with. By including/excluding the new target t_c, we obtain two target sets $(t_c, t_{i1}, t_{i2}, ..., t_{in_i})$ and $(t_{i1}, t_{i2}, ..., t_{in_i})$ respectively. For each target set, we calculate the set of pairwise common substrings and recover two substring sets, the *withSS* set $(s_{i1}, s_{i2}, ..., s_{ip'})$ and the *withoutSS* set $(s_{i1}, s_{i2}, ..., s_{iq'})$ from the two target sets respectively. Note that if the active region (i.e., the drug binding region) of the new target is contained in at least 2 targets that interact with d_i, then it will still appear in the $withoutSS$ substring set. However, the $withSS$ substring set could be more important because it increases the chances that any key active region is included in the substring sets. Although the $withSS$ set increases

the number of unimportant substrings to be included in the substring sets, this issue will be addressed in the automated feature selection model shown below. Because a substring may appear more than once in the *withSS* or *withoutSS* set, and some substrings might be very similar, we cluster the substrings to obtain a reduced *withSS* set $(s_{i1}, s_{i2}, ..., s_{ip})$ and reduced *withoutSS* set $(s_{i1}, s_{i2}, ..., s_{iq})$, additionally recording the number of occurrences of each substring. For the *withSS* set, we vectorize the training targets and the new coming target t_c by computing the match scores between their full sequences and each substring in the reduced *withSS* set. For a target t and a feature substring s, the normalized match score is calculated as:

$$M(t, s) = \frac{L(t,s) \cdot c_s^d}{\sum_{j=1}^{p} c_{s_i}^d}, \tag{3.12}$$

where $L(\cdot, \cdot)$ is length of the longest common substring between the two sequences, c_s is the number of occurrence of substring s, and d is a constant used to increase the importance of more frequent substrings. In practice we set $d = 1$ after trying $d = 1, .., 4$. Thus, for the normalized match score $M(\cdot, \cdot)$, if the target contains a substring that is long and occurs multiple times in the reduced substring feature set, the score for this feature substring will be high. We then obtain an $n \times p$ training matrix X and a $p \times 1$ testing vector $\mathbf{x_c}$, along with the $n \times 1$ binary training label vector (with 1 indicating the target interacts with the drug and -1 otherwise). The problem is now formulated as a standard classification problem. After classifying the new target, 1 will indicate that it interacts with the drug d_i, and -1 will indicate otherwise. The same procedure is repeated on the $withoutSS$ substring set.

Classification with Feature Selection In any classification problem, the quality of features used determines the accuracy of predictions. Here, features correspond to substrings of target proteins, which comprise potential binding regions between the proteins and drugs. Thus, selecting good features not only improves classification accuracy, but also provides candidate drug-target binding sites for further investigation.

I first investigated a feature selection SVM model developed based on classic L_2 norm regularized SVM. The primal form of a classic SVM problem can be

formulated as follows:

$$\begin{array}{c} \min\limits_{\mathbf{w}} \dfrac{\beta_1}{2}\mathbf{w}^T\mathbf{w} + \mathbf{1}^T\xi, \\ \textbf{s.t.} : \xi \geq \mathbf{1} - \triangle(\mathbf{y})X\mathbf{w}, \\ \xi \geq 0. \end{array} \tag{3.13}$$

Its dual form is:

$$\begin{array}{c} \max\limits_{\lambda} \lambda^T\mathbf{1} - \dfrac{1}{2\beta_1}\lambda^T\triangle(\mathbf{y})XX^T\triangle(\mathbf{y})\lambda, \\ \textbf{s.t.} : 0 \leq \lambda \leq 1. \end{array} \tag{3.14}$$

Similar to [51], I introduce the feature selection vector $\gamma \geq 0$ and letting $\tilde{X} = X\triangle(\gamma)^{1/2}$. Let $\tilde{K} = \tilde{X}\tilde{X}^T = X\triangle(\gamma)X^T = \sum_j \gamma_j X_{:j}X_{:j}^T = \sum_j \gamma_j K_j$, we get:

$$\min_{\gamma\geq 0} \max_{0\leq\lambda\leq\mathbf{1}} \lambda^T\mathbf{1} - \frac{1}{2\beta_1}\lambda^T\triangle(\mathbf{y})(\sum_j \gamma_j K_j)\triangle(\mathbf{y})\lambda + \beta_2\mathbf{1}^T\gamma, \tag{3.15}$$

Equation (3.15) can be reformulated and solved in the three approaches, i.e., semi-definite programming, quadratic constraint linear programming (QCLP), and generic constraint generation, described in Appendix D. After implementation and testing, we found none of these three approaches are either effective or efficient. The inefficiency of the three approaches is due to the nature of the three approaches and time spent on training two induced parameters β_1 and β_2. The ineffectiveness is due to the lack of kernelization that are usually adopted in classic SVM classifiers. To circumvent these disadvantages, we then investigated an approach that integrates feature selection in L_1-norm based support vector machine (SVM) classification method.

The primal form of L_1-norm SVM is:

$$\begin{array}{rc} & \min\limits_{\mathbf{w},\mathbf{b},\xi} \beta\|\mathbf{w}\|_1 + \mathbf{1}^T\xi \\ \textbf{s.t.} : & \xi \geq \mathbf{1} - \triangle(\mathbf{y})(X\mathbf{w} - \mathbf{1}b), \\ & \xi \geq \mathbf{0}. \end{array} \tag{3.16}$$

Here $X \in \mathbb{R}^{n\times p}$, $\mathbf{y} \in \mathbb{R}^n$, n is the number of data points, and p is the number of features. The value of β affects the sparsity level of $\mathbf{w}$ and if prior knowledge is available, an appropriate β can be chosen beforehand. Since

$$\|\mathbf{w}\|_1 = \min_{\gamma\geq 0} \frac{1}{2}\sum_j (\frac{w_j^2}{\gamma_j} + \gamma_j) = \min_{\gamma\geq 0} \frac{1}{2}(\mathbf{w}^T G^{-1}\mathbf{w} + \gamma^T\mathbf{1}) \tag{3.17}$$

[57], where $G = \triangle(\gamma)$, (3.16) becomes

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b},\xi,\gamma} \ & \frac{\beta}{2}(\mathbf{w}^T G^{-1}\mathbf{w} + \gamma^T \mathbf{1}) + \mathbf{1}^T \xi \\ \textbf{s.t.}: \ & \xi \geq \mathbf{1} - \triangle(\mathbf{y})(X\mathbf{w} - \mathbf{1}b), \\ & \xi \geq \mathbf{0}, \gamma \geq \mathbf{0}. \end{aligned} \tag{3.18}$$

By introducing Lagrangian multipliers $\lambda \geq \mathbf{0}$ and $\mu \geq \mathbf{0}$, (3.18) becomes

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b},\xi,\gamma} \max_{\lambda,\mu} & \frac{\beta}{2}(\mathbf{w}^T G^{-1}\mathbf{w} + \gamma^T \mathbf{1}) + \mathbf{1}^T \xi \\ & + \lambda^T(\mathbf{1} - \triangle(\mathbf{y})(X\mathbf{w} - \mathbf{1}b) - \xi) - \mu^T \xi \\ & \textbf{s.t.}: \quad \lambda \geq \mathbf{0}, \mu \geq \mathbf{0}. \end{aligned} \tag{3.19}$$

Let the objective function of (3.19) be L_1, and let $\frac{\partial L_1}{\partial \gamma} = 0$, we get $\lambda = \mathbf{1} - \mu$. Therefore, since $\mu \geq \mathbf{0}$, we conclude that $\lambda \leq \mathbf{1}$, hence $\mathbf{0} \leq \lambda \leq \mathbf{1}$. By substitution, (3.19) becomes

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b},\gamma} \max_{\lambda} & \frac{\beta}{2}(\mathbf{w}^T G^{-1}\mathbf{w} + \gamma^T \mathbf{1}) + \lambda^T \mathbf{1} \\ & -\lambda^T \triangle(\mathbf{y}) X\mathbf{w} + \lambda^T \triangle(\mathbf{y})\mathbf{1}b \\ & \textbf{s.t.}: \quad \mathbf{0} \leq \lambda \leq \mathbf{1}. \end{aligned} \tag{3.20}$$

Let the objective function of (3.20) be L_2, and let $\frac{\partial L_2}{\partial b}$ = 0. We get $\lambda^T \mathbf{y} = 0$, so (3.20) becomes

$$\begin{aligned} \min_{\mathbf{w},\gamma} \max_{\lambda} & \frac{\beta}{2}(\mathbf{w}^T G^{-1}\mathbf{w} + \gamma^T \mathbf{1}) + \lambda^T \mathbf{1} - \lambda^T \triangle(\mathbf{y}) X\mathbf{w} \\ & \textbf{s.t.}: \quad \mathbf{0} \leq \lambda \leq \mathbf{1}, \\ & \lambda^T \mathbf{y} = 0. \end{aligned} \tag{3.21}$$

Let the objective function of (3.21) be L_3, and let $\frac{\partial L_3}{\partial \mathbf{w}} = 0$, we get $\beta G^{-1}\mathbf{w} - X^T \triangle(\mathbf{y})\lambda = \mathbf{0}$, so that $\mathbf{w} = \frac{1}{\beta} G X^T \triangle(\mathbf{y})\lambda$. By substitution, (3.21) becomes

$$\begin{aligned} \min_{\gamma} \max_{\lambda} & \lambda^T \mathbf{1} - \frac{1}{2\beta}\lambda^T \triangle(\mathbf{y}) X G X^T \triangle(\mathbf{y})\lambda + \frac{\beta}{2}\gamma^T \mathbf{1} \\ & \textbf{s.t.}: \quad \mathbf{0} \leq \lambda \leq \mathbf{1}, \\ & \lambda^T \mathbf{y} = 0, \\ & \gamma \geq \mathbf{0}. \end{aligned} \tag{3.22}$$

Note that γ is the feature selection vector. Crucially, this problem is convex in μ and has no local minima [51], hence it provides an optimal form of feature selection that can be efficiently obtained in conjunction with SVM training. Because a drug may bind to different regions of different proteins, i.e., different regions on

different targets can bind to the same drug, each positive data points may correspond to a different set of important features (substrings). Therefore, the nature of this drug-target classification problem is essentially a multi-instance classification problem. To address this, we consider two tricks: (a) Use a radial basis function (RBF) kernel (Gaussian kernel), rather than a linear kernel since this addresses the multi-instance classification problem more effectively after mapping data points to an implicit infinite dimensional space. After Gaussian kernelization, the original linear kernel matrix $K = XGX^T$ becomes $K'_{ij} = e^{\frac{-1}{2\sigma^2}(\mathbf{x_i}-\mathbf{x_j})\triangle(\gamma)(\mathbf{x_i}-\mathbf{x_j})^T}$. (b) Because each positive data point may correspond to a unique set of important features, in principle, N^+ number of positive γ vectors and one negative γ vector should be considered where N^+ is the number of positive data points in the training set. So we get $K''_{ij} = e^{\frac{-1}{2\sigma^2}\|\mathbf{x_i}.*\gamma_\mathbf{i}-\mathbf{x_j}.*\gamma_\mathbf{j}\|^2}$, $\forall i, j$, where $\gamma_\mathbf{i} = \gamma_\mathbf{i}^+$ if $y_i = +1$, and $\gamma_\mathbf{i} = \gamma^-$ if $y_i = -1$ (same for $\gamma_\mathbf{j}$). Trick (a) can be easily applied on (3.22) although it may damage its convexity, while applying (b) into (3.22) will introduce too many extra coefficients, corresponding to each $\gamma_\mathbf{i}^+$ and make the model computationally expensive. To circumvent this, we introduce a feature cost vector $\mathbf{c} \in \mathbb{R}^p$, where $c_j = \frac{1}{n}\sum_i a_{ij}$, $a_{ij} = 1$ if $x_{ij} \geq y'_i - \epsilon$ and $a_{ij} = 0$ if $x_{ij} < y'_i - \epsilon$, where ϵ is a small value. Note that for computing $\mathbf{c}$, each column of X will be normalized to to [0,1], and $y'_i = 0$ if $y_i = -1$. The intuition of introducing $\mathbf{c}$ is to penalize the features that are false positive indicator of binding, which is exactly the case in multi-instance learning where true positive indicators is not so much concerned while false positive needs to be strictly controlled. Therefore, we use $\tilde{X} = X \cdot \triangle(\mathbf{1}-\mathbf{c})$ in (3.22) to encourage features that are less false positive indicators to be selected, which yields

$$\begin{aligned} \min_{\gamma} \max_{\lambda} \lambda^T\mathbf{1} - \frac{1}{2\beta}\lambda^T\triangle(\mathbf{y})K'\triangle(\mathbf{y})\lambda + \frac{\beta}{2}\gamma^T\mathbf{1} \\ \textbf{s.t.}: \quad \mathbf{0} \leq \lambda \leq \mathbf{1}, \\ \lambda^T\mathbf{y} = 0, \\ \gamma \geq \mathbf{0}, \end{aligned} \tag{3.23}$$

where here $K'_{ij} = e^{\frac{-1}{2\sigma^2}(\tilde{\mathbf{x}}_\mathbf{i}-\tilde{\mathbf{x}}_\mathbf{j})\triangle(\gamma)(\tilde{\mathbf{x}}_\mathbf{i}-\tilde{\mathbf{x}}_\mathbf{j})^T}$.

We solve (3.23) using a combination of LBFGS and gradient decent method over γ. After optimization, we get solutions for γ and λ, where a $\gamma_j > \epsilon$ indicates the j's features should be selected and otherwise not. λ is used to construct the

hyperplane in the SVM and predict new data points. So for a testing data point $\mathbf{x}'$, we have $\hat{y}' = \sum_i \lambda_i y_i K(\tilde{\mathbf{x}}_\mathbf{i} \triangle(\gamma), \tilde{\mathbf{x}}' \triangle(\gamma)) - b$ (by removing the constraint $\lambda^T \mathbf{y} = 0$, b = 0). As a key step for solving (3.23), we demonstrate the derivation of the partial derivatives with respect to γ. Let the objective function of (3.23) be L_4, the partial derivative of L_4 with respect to the k's feature γ_k is

$$\frac{\partial L_4}{\partial \gamma_k} = \frac{1}{2\beta} \sum_{ij} \lambda_i \lambda_j y_i y_j \frac{\partial K'_{ij}}{\partial \gamma_k} + \frac{\beta}{2},$$

where

$$\frac{\partial K'_{ij}}{\partial \gamma_k} = K'_{ij} \Big[\frac{-1}{2\sigma^2} (\tilde{x}_{ik} - \tilde{x}_{jk})^2 \Big],$$

$$= \quad e^{\frac{-1}{2\sigma^2} (\tilde{\mathbf{x}}_\mathbf{i} - \tilde{\mathbf{x}}_\mathbf{j}) \triangle(\gamma) (\tilde{\mathbf{x}}_\mathbf{i} - \tilde{\mathbf{x}}_\mathbf{j})^T} \Big[\frac{-1}{2\sigma^2} (\tilde{x}_{ik} - \tilde{x}_{jk})^2 \Big].$$

3.2.2 Experimental Results

Datasets

We used drug-target interaction information from [7], which was collected from the KEGG BRITE [42], BRENDA [69], SuperTarget [24] and DrugBank [89] databases. In particular, we used three data sets—nuclear receptors, GPCRS, and ion channel—which have (54, 223, 210) drugs, (26, 95, 204) targets, and (90, 635, 1476) interactions, respectively. The three data sets used in this dissertation are identical to the three used in the state of the art study [7], to facilitate a comparison between the two methods. Since we only focus on target-side prediction, we do not require any drug structural or pharmacological information to obtain drug-drug similarity information. The amino acid sequences of the target proteins were obtained from the KEGG GENES database [42].

Comparison methods

We compare our method to the state of the art method proposed by Bleakley et al. [7]. In particular, we only focus on target-side prediction of their method to make the two methods comparable. The framework of the two methods is similar, except Bleakley et al. use the normalized Smith-Waterman score in (3.11) to evaluate the similarity between two target sequences. When using an SVM classifier, the target-target similarity matrix is considered as a kernel matrix for classification, hence the

kernel matrix is fixed in their method. In addition to the SVM classification method, we also used the nearest neighbor (NN) classifier for both methods as a baseline. We refer to Bleakley et al.'s approach as BLM_SVM and BLM_NN respectively. We refer to the *withSS* feature set based L1-SVM (the main model of this chapter), SVM (the classic L2 norm SVM) and nearest neighbor classification methods as SS_L1-SVM, SS_SVM, SS_NN_FS, and SS_NN_noFS respectively, where SS_NN_FS is the nearest neighbor classification based on the features selected by the SS_L1-SVM method and SS_NN_FS is the nearest neighbor classification based on all features.

Classification results We use five measurements to evaluate the quality of drug-target prediction: Area under the Precision-Recall Curve (AUPR), Area under the ROC Curve (AUC), F-Measure, Precision, and Recall. Of the five measurements, AUPR, AUC, and F-Measure are more robust measurements than the others. For each prediction method, the Precision and Recall scores are obtained at the cutoff point where F-Measure is maximized. Tables 3.3, 3.4, and 3.5 demonstrate the effectiveness of the different drug-target prediction methods over the five evaluation quantities. The training data are selected so that each drug has at least 2 binding targets (i.e., degree at least 2). Figure 3.5 demonstrate the precision-recall curves of SS_L1-SVM and SS_SVM compared to BLM_SVM, BLM_NN, SS_NN_FS, and SS_NN_noFS on the Nuclear, GPCR, and Ion Channel data sets, respectively.

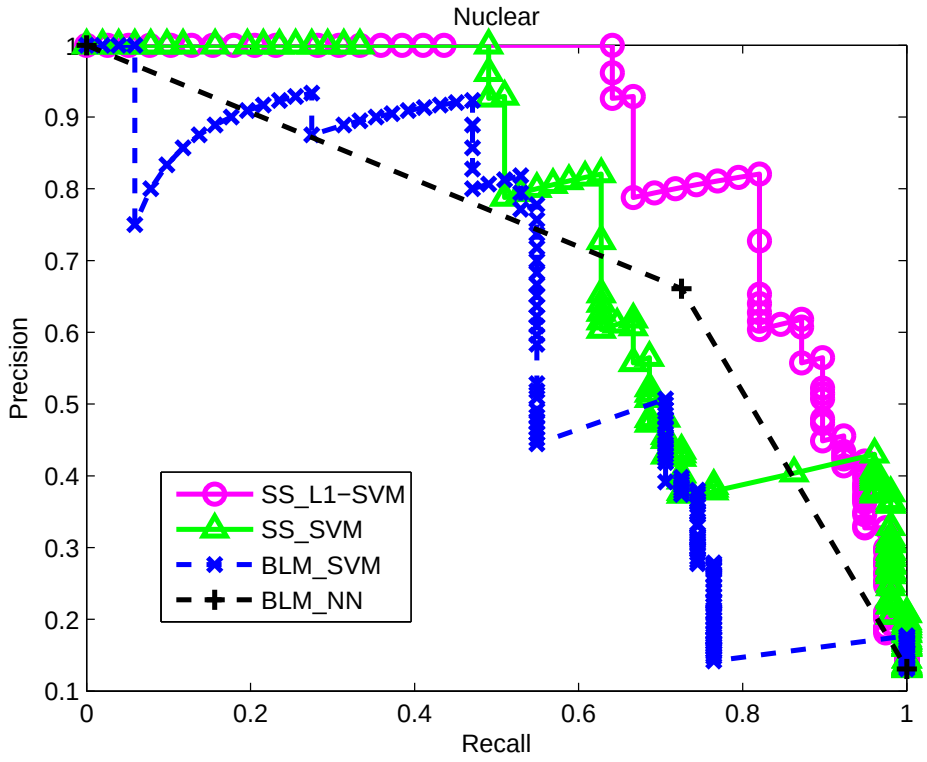


Figure 3.3: The precision-recall curves of the four methods SS_L1-SVM, SS_SVM, BLM_SVM, BLM_NN, SS_NN_FS, SS_NN_noFS on Nuclear dataset. The results are based on training data with drug interacting with at least 2 targets.

Table 3.3: Evaluations of classification quality on Nuclear data set. For each prediction method, the Precision and Recall scores are obtained at the cutoff point where F-Measure is maximized. The results are based on training data with drug interacting with at least 2 targets.

Performance comparison	AUPR	AUC	FMeasure	Precision	Recall
SS_L1-SVM	**0.8756**	**0.9512**	**0.8205**	**0.8205**	0.8205
SS_SVM	0.7635	0.9277	0.7111	0.8205	0.6275
BLM_SVM	0.6163	0.8034	0.6353	0.7941	0.5294
BLM_NN	0.7111	0.8347	0.6916	0.6607	0.7255
SS_NN_FS	0.6985	0.8680	0.6415	0.5075	**0.8718**
SS_NN_noFS	0.6743	0.8459	0.6308	0.5190	0.8039

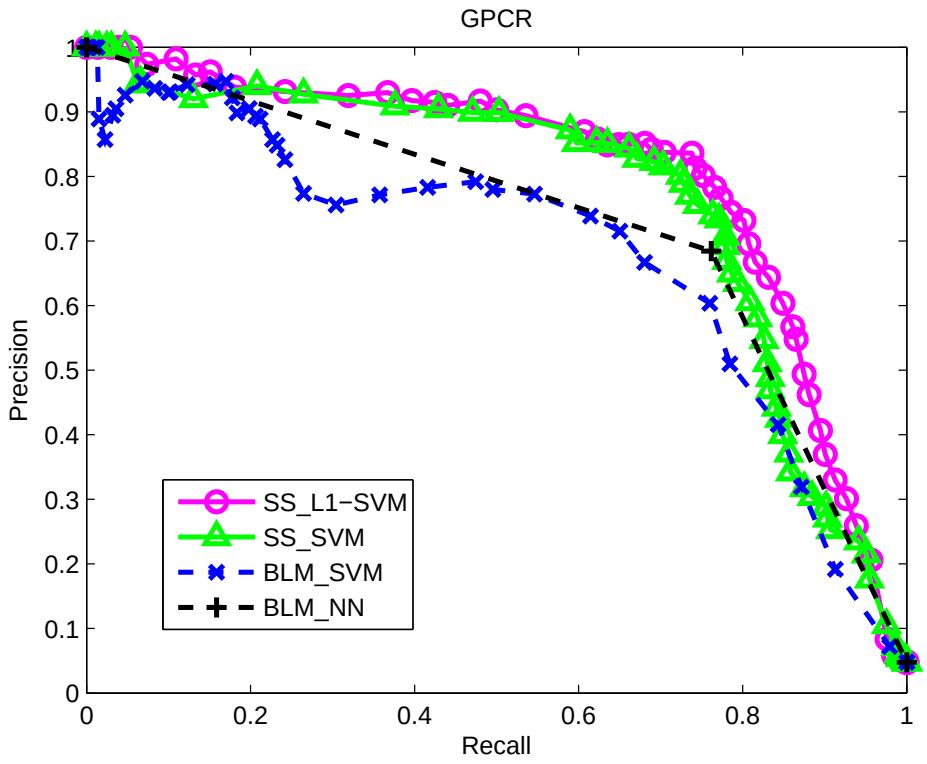


Figure 3.4: The precision-recall curves of the four methods SS_L1-SVM, SS_SVM, BLM_SVM, BLM_NN, SS_NN_FS, SS_NN_noFS on GPCR dataset. The results are based on training data with drug interacting with at least 2 targets.

Table 3.4: Evaluations of classification quality on GPCR data set. For each prediction method, the Precision and Recall scores are obtained at the cutoff point where F-Measure is maximized. The results are based on training data with drug interacting with at least 2 targets.

Performance comparison	AUPR	AUC	FMeasure	Precision	Recall
SS_L1-SVM	**0.803**9	**0.9603**	**0.7840**	**0.8360**	0.7381
SS_SVM	0.7720	0.9600	0.7607	0.8013	0.7240
BLM_SVM	0.6800	0.9435	0.6812	0.7152	0.6503
BLM_NN	0.7287	0.8721	0.7209	0.6842	0.7618
SS_NN_FS	0.7155	0.8878	0.6997	0.6219	**0.7996**
SS_NN_noFS	0.7219	0.8875	0.7081	0.6365	0.7977

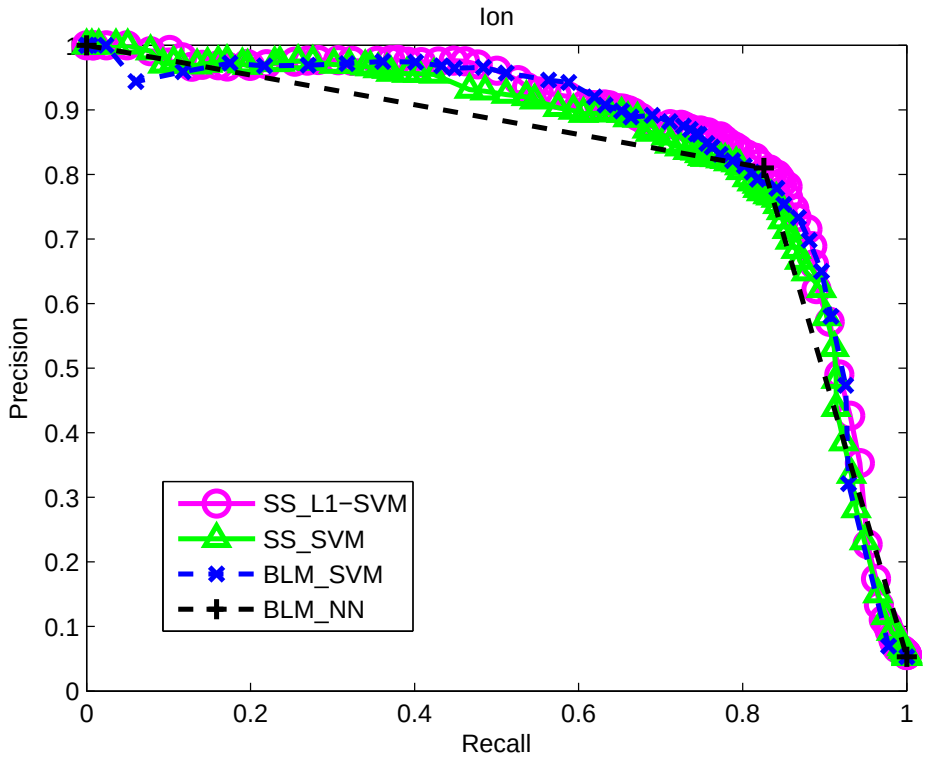


Figure 3.5: The precision-recall curves of the four methods SS_L1-SVM, SS_SVM, BLM_SVM, BLM_NN, SS_NN_FS, SS_NN_noFS on Ion dataset. The results are based on training data with drug interacting with at least 2 targets.

Table 3.5: Evaluations of classification quality on Ion data set. For each prediction method, the Precision and Recall scores are obtained at the cutoff point where F-Measure is maximized. The results are based on training data with drug interacting with at least 2 targets.

Performance comparison	AUPR	AUC	FMeasure	Precision	Recall
SS_L1-SVM	**0.8632**	0.9666	**0.8205**	**0.8260**	0.8151
SS_SVM	0.8450	**0.9690**	0.8045	0.8173	0.7921
BLM_SVM	0.8561	0.9568	0.8088	0.7785	**0.8416**
BLM_NN	0.8226	0.9075	0.8179	0.8101	0.8258
SS_NN_FS	0.7041	0.8542	0.6954	0.6647	0.7290
SS_NN_noFS	0.6702	0.8640	0.6497	0.5671	0.7606

Based on these evaluations, one can observe that the *withoutSS* feature set based SVM approaches, i.e., SS_L1-SVM and SS_SVM outperform the others on all three

datasets, including the current state of the art methods BLM_SVM and BLM_NN. Moreover, the L1 norm feature selection method SS_L1-SVM is more effective than the traditional SVM method and it only use 72.85%, 85.02%, and 62.86% numbers of features our of the original feature sets on Nuclear, GPCRs, and Ion Channels datasets, respectively. The significant reduction in feature set size cannot only make the classification more efficient and effective, it can also help one to more accurately identify important features.

After investigating the prediction results generated by the SS_L1-SVM and BLM_SVM methods, at the prediction cutoff where both methods attain their maximum FMeasure score, we find that there are 8, 127, and 78 true positive interactions that can be identified by the SS_L1-SVM method but cannot be identified by the BLM_SVM method (compared with 7, 16, 52 true positives that can be identified by the BLM_SVM method but cannot be identified by the SS_L1-SVM method).

For example, on the Nuclear dataset, according to the normalized Smith-Waterman scores, the two nearest neighbors of target protein RORB (KEGG Homo sapiens protein ID "hsa6096") are RORA ("hsa6095") and RORC ("hsa6097"), with scores 0.578 and 0.458 respectively. RORB and RORC share a common interacting drug *Tretinoin* (KEGG drug ID "D00094") while RORB and RORA do not. Using the BLM method, when predicting the interactions of target RORB, it is predicted to have no interaction with *Tretinoin*, since its nearest neighbor RORA does not interact with *Tretinoin*. On the contrary, our method can correctly identify the interaction between RORB and *Tretinoin* because the withSS feature set based method can discover two important substrings "EVVLVRMCRA-N" and "N-TV-FEGKYGGM" that exist in both RORB and RORC. Therefore, although the overall match score between RORB and RORC is not the highest, their scores feature vectors with respect to the two feature substrings are the most similar pair among all the others.

On the GPCR dataset, according to the normalized Smith-Waterman scores, the 5 nearest neighbors of the target protein CHRM1 (KEGG Homo sapiens protein ID "hsa1128") are CHRM5 ("hsa1133"), CHRM3 ("hsa1131"), CHRM4 ("hsa1132"), CHRM2 ("hsa1129"), and HRH3 ("hsa11255"), with scores 0.4707, 0.4536, 0.4237, 0.4228, and 0.2446 respectively. CHRM1 is supposed to bind drug *Metoclopramide*

(KEGG drug ID "D00726"), but none of its 5 nearest neighbors bind to this drug until its 6'th nearest neighbor HRH2 ("hsa3274") with SW_{norm} score 0.2137. Therefore, in the BLM methods, CHRM1 is not predicted to bind *Metoclopramide*. On the contrary, our method can correctly predict the interaction between CHRM1 and *Metoclopramide* because the important substrings "KRTPRRAA", "Y-AKRTP-RAA-MI-L-W", "WL-Y-NS-INP", "NYFL-SLA-AD", and "PL-YR-K-TP-R-ALM-G" are present in both CHRM1 and the proteins that binds to *Metoclopramide*, e.g., HTR1A ("hsa3350"), HTR1B ("hsa3351"), HTR1D ("hsa3352"), HTR1E ("hsa3354"), HTR1F ("hsa3355"), HTR2A ("hsa3356"), HTR2B ("hsa3357"), HTR2C("hsa3358"), HTR4("hsa3360"), HTR5A("hsa3361"), and HTR6("hsa3362"), which are all considered as far away neighbors based on the SW_{norm} scores. These kind of observation can be found across all the three datasets.

3.2.3 Discussion

In this section, we proposed a novel drug-target interaction prediction method based on identifying potential drug-target binding regions. According to the evaluation metrics, the proposed method significantly outperforms the current state-of-the-art methods on this task. More importantly, the proposed method can identify a number of drug-target interactions that were missed by previous methods. We believe that the poor recall of previous methods is due to the use of a target kernel matrix based on Smith-Waterman score: a low overall similarity between two protein sequence sequences does not mean they do not share common drug binding regions. This drawback is avoided in our approach by collecting a large number of candidate binding regions (i.e., common substrings) that subsequently play the primary role in interaction prediction. In addition, the use of an explicit vector representation, as opposed to implicit similarity measure, enables the easy use of non-linear kernel expansions that are not possible for fixed kernel methods like BLM.

We presented a feature selection L1-norm SVM based method that can not only predict the binding relations more accurately, but also find important candidate binding regions (features). It integrated feature selection directly into an L_1-norm SVM and kernelized the optimization model. A key drawback was that the sparse

regularization term tended to select only a single feature from the candidate set. This is a well known problem with L_1 based regularization. To avoid this limitation, I will investigate a combination of L_1 and L_2 norm regularizers, known as the elastic net [98], which is generally more effective at group feature selection. Another possible extension is to adopt the OSCAR model [95], which appears even more effective (but is also more expensive). We also discovered that the drug-target interaction inference problem—in some cases—can be considered as multi-instance learning problem, so we proposed to use multiple feature selection vectors for each positive training data points in theory and applied the feature cost vector to address the multi-instance problem in practice. We hypothesize that more advanced machine learning methods specifically tailored for multi-instance classification can further improve the accuracy of drug-target interaction prediction.

Chapter 4

Conclusion

This thesis has investigated some important problems in the bio-relation discovery (BRD), while pursuing a novel methodology for applying sparse learning optimization techniques in these problems, in comparison to some initial heuristic approaches. The sparse learning methods share a common framework that sparse regularizers are introduced to select important features, while additionally kernelization may be applied. In particular, I have categorized the BRD area into to subareas, undirected BRD and directed BRD. In undirected BRD, I presented the gene expression bi-clustering problem and investigated three approaches (two heuristic approaches and one sparse learning based approach). For directed BRD, I presented two problems, the genetic regulatory network problem and the drug-target network inference problem. For both of these problems, sparse learning models are developed and kenelization approaches are further investigated.

4.1 Summary of Contributions

In Chapter 2, to solve the linear coherent bi-clustering problem, I first proposed two heuristic approaches, the line detection based LCBD and the beam detection based LinCoh. LCBD uses the Hough transform [33] technique, that was originally developed in computer graphics, to find gene pairwise linear coherence while LinCoh uses beam detection to find gene pairwise linear coherence. In the second step of LCBD and LinCoh, a sample majority voting and sample set clustering are used respectively to do achieve final bi-clustering. LCBD and LinCoh both identify linear-

coherent bi-clusters that disclose undirected relations between genes, while LinCoh is a more practical algorithm on microarray datasets due to data noise. So LinCoh can serve as another useful tool for microarray data analysis, including bi-clustering and genetic regulatory network inference. After LCBD and LinCoh, I proposed another novel approach SLLB, that can discover linear coherent bi-clusters based on a sparse learning optimization model. The time complexity of the SLLB algorithm is $O(n^2k)$ where n is the number of observations and k is the number of iterations that SLLB takes to converge, which is very fast comparing to algorithms like LinCoh. Note that while discovering linear coherent bi-clusters, SLLB favors data points corresponding to features that are far away from each other in the observation pair 2D space; this nice property can be used for downstream data analysis such as feature clustering, observation-feature relation studies and observation/feature selection.

To address the genetic regulatory network problem in Chapter 3, I proposed a new kernelized version of globally regularized risk minimization objective for learning regulatory networks from gene expression data. Exploiting the assumption that genes with similar expression patterns are likely to be co-regulated, I first clusters the genes, then learn the regulatory relationships by encouraging genes with similar expression patterns to share regulators. I kernelize the linear model to map the independent linear relations between transcription factors to more complex relations in high dimensional space, considering in real cases, the TF-gene regulation rules likely obey the so-called canalyzing rules. I conjecture that the kernelized model can avoid the problem of enumerating all the canalyzing regulations and can discover the non-linear relations between transcription factors to better construct the target gene profile and more accurately infer the TF-gene causal relations. To address the regulation time lag problem, I proposed to use cubic spline interpolation to represent the discrete gene profiles to continuous profiles and look for the potential time lag between transcription factor and a set of similar genes instead of a single gene. This makes the time lag searching more robust to noise. I also considered the within-cluster gene competition effect that is neglected by most GRN inference methods in the literature.

In Chapter 3, I also proposed a novel drug-target interaction prediction method based on identifying potential drug-target binding regions. Besides its effectiveness in predicting drug-target interactions, the proposed method can identify a number of drug-target interactions that were missed by previous methods. It is believed that the poor recall of previous methods is due to the use of a target kernel matrix based on Smith-Waterman score: a low overall similarity between two protein sequence sequences does not mean they do not share common drug binding regions. This drawback is avoided in our approach by collecting a large number of candidate binding regions (i.e., common substrings) that subsequently play the primary role in interaction prediction. In addition, the use of an explicit vector representation, as opposed to implicit similarity measure, enables the easy use of non-linear kernel expansions that are not possible for fixed kernel methods like BLM. The approach integrates feature selection directly into L_1-norm SVM and kernelize the optimization model.

The experimental results demonstrate that the sparse learning approach cannot only infer different types of bio-relations more accurately, but also generate important information that reveals the biological reasons that cause the bio-relations.

4.2 Future Research Directions

A disadvantage of LinCoh are its large memory and compute time requirements, due to constructing the outer and inner sample set matrices. It takes $O(n^2 m_p)$ to compute the sample set matrices where n is the number of genes and m_p is the number of parameters θ, β and γ. The memory required for storing the matrices is $O(n^2 m_s)$ where n is the number of genes and m_s is the average size of the sample set elements. It takes weeks and up to 1 Gigabyte memory to run experiments on the E.coli dataset. Improvements in beam detection and sample set clustering can also achieve significant speed-ups.

For SLLB, although the two parameters β_1 and β_2 in our synthetic experiments are selected in a way that give good results, it is quite robust to different settings. In both synthetic data and real data experiments, I did not try many β_1 β_2 settings

but found the results were all very good under different evaluations. When comes to practice, considering that β_1 actually controls the size of observation and β_2 controls the size of features in the result bi-clusters, β_1 and β_2 can be determined when prior knowledge of bi-cluster size is known. I suggest that the SLLB algorithm can be used in other machine learning applications such as image clustering, document clustering, and other biology and health care data clustering, as long as observations of the same group have linear coherence under a subset of features, and for different clusters, different feature sets need to be selected. It may be useful to test SLLB on other applications such as document bi-clustering and image bi-clustering. Extensions of SLLB that consider other relations between observations in addition to the linear coherent relations is also worth investigating.

For the GRN problem, further prediction improvements are likely to come from incorporating further sources of biologically relevant data, such as the within-cluster gene competition mRNA upper bound, binding information [76], or other forms of prior knowledge beyond the co-regulation assumption made here. Moreover, as an effective strategy, the kernel method combined with the $L1$ norm feature selection might be extended to resolve other similar problems in bioinformatics area and other research areas.

For the drug-target interaction inference problem, a key drawback was that the sparse regularization term tended to select only a single feature from the candidate set. This is a well known problem with L_1 based regularization. To avoid this limitation, one can investigate a combination of L_1 and L_2 norm regularizers, known as the elastic net [98], which is generally more effective at group feature selection. Another possible extension is to adopt the OSCAR model [95], which appears even more effective. It is also discovered that the drug-target interaction inference problem—in some cases—can be considered as multi-instance learning problem, so I propose to use multiple feature selection vectors for each positive training data points in theory and applied the feature cost vector to address the multi-instance problem in practice. I conjecture that more advanced machine learning methods specifically tailored for multi-instance classification can further improve the accuracy of drug-target interaction prediction.

Bibliography

[1] M. Ashburner, C. A. Ball, and J. A. Blake *et. al*. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[2] W. Ayadi, M. Elloumi, and J. K. Hao. Pattern-driven neighborhood search for biclustering of microarray data. *BMC Bioinformatics*, 13:(Suppl 7):S11, 2012.

[3] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving sub-matrix problem. In *RECOMB'02*, pages 49–57, 2002.

[4] A. Bernard and A. Hartemink. Informative structure priors: joint learning of dynamic regulatory netwokrs from multiple types of data. *Pac. Symp. Biocomput.*, pages 459–470, 2005.

[5] G. F. Berriz, O. D. King, B. Bryant, C. Sander, and F. P. Roth. Characterizing gene sets with FuncAssociate. *Bioinformatics*, 19:2502–2504, 2003.

[6] D. Bertsekas. Nonlinear optimization. *Athena Scientific*, 1995.

[7] K. Bleakley and Y. Yamanishi. Supervised prediction of drugctarget interactions using bipartite local models. *Bioinformatics*, 25:2397–2403, 2009.

[8] S. Boyd and L. Vandenberghe. Convex optimization. *Combridge Univ. Press*, 2004.

[9] Z. Cai, L. Xu, Y. Shi, M.R. Salavatipour, R. Goebel, and G. Lin. Using gene clustering to identify discriminatory genes with higher classification accuracy. *IEEE Symposium on Bioinformatics and Bioengineering*, pages 235–242, 2006.

[10] M. Campillos1, M. Kuhn, A. C. Gavin, and et al. Drug target identification using side-effect similarity. *Science*, 321:263–266, 2008.

[11] H. C. Causton, J. Quackenbush, and A. Brazma. Microarray gene expression data analysis : A beginner's guide. *Blackwell Publishing, Malden*, 2003.

[12] K.C. Chen, T.Y. Wang, H.H. Tseng, C.Y.F. Huang, and C.Y. Kao. A stochastic differential equation model for quantifying transcriptional regulatory network in saccharomyces cerevisiae. *Bioinformatics*, 21:2883–2890, 2005.

[13] X. Chen, G. Anantha, and X. Wang. An effective structure learning method for constructing gene networks. *Bioinformatics*, 22:1367–1374, 2006.

[14] A. Cheng, R. Coleman, K. Smyth, and et al. Structure-based maximal affinity model predicts small-molecule druggability. *Nat. Biotechnol.*, 25:71–75, 2007.

[15] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[16] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, and R.W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, 2:65–73, 1998.

[17] P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cns development and injury. *Pac. Symp. Biocomput.*, pages 41–52, 199.

[18] C. M. Dobson. Chemical space and biology. *Nature*, 432:824–828, 2004.

[19] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96*, pages 226–231, 1996.

[21] J. J. Faith, M. E. Driscoll, and V. A. Fusaro *et al.* Many microbe microarrays database: uniformly normalized Affymetrix compendia with structured experimental metadata. *Nucleic Acids Research*, 36:D866–D870, 2008.

[22] X. Gan, A. W-C. Liew, and H. Yan. Discovering biclusters in gene expression data based on high-dimensional linear geometries. *BMC Bioinformatics*, 9:209, 2008.

[23] A. P. Gasch, P. T. Spellman, and C. M. Kao *et al.* Genomic expression programs in the response of yeast cells to environmental changes. *Nucleic Acids Research*, 11:4241–4257, 2000.

[24] S. Gunther, M. Kuhn, M. Dunkel, and et al. Supertarget and matador: Resources for exploring drug-target relationships. *Nucleic Acids Res.*, 36:D919–22, 2008.

[25] Y. Guo and D. Schuurmans. Convex structure learning for bayesian netwokrs: Polynomial feature selection and approximating ordering. *Conf. on Uncertainty in Artif. Intell*, pages 208–216, 2006.

[26] Y. Guo and D. Schuurmans. Learning gene regulatory networks via globally regularized risk minimization. *RECOMB Satellite Workshop on Comparative Genomics (LNCS)*, 4751:83–95, 2007.

[27] R. Gupta and V. Kumar N. Rao. Discovery of error-tolerant biclusters from noisy gene expression data. *Bioinformatics*, 12:(Suppl 12):S1, 2011.

[28] S. Haggarty, K. Koeller, J. Wong, and et al. Multidimensional chemical genetic analysis of diversity-oriented synthesis-derived deacetylase inhibitors using cell-based assays. *Chem. Biol.*, 10:383–396, 2003.

[29] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pac. Symp. Biocomput*, pages 422–444, 2001.

[30] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67:123–129, 1972.

[31] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[32] M.J.L. De Hoon, S. Imoto, K. Kobayashi, N. Ogasawara, and S. Miyano. Inferring gene regulatory networks from time-ordered gene expression data of bacillussubtilis using differential equations. *Pac. Symp. Biocomput.*, pages 17–28, 2003.

[33] P.V.C. Hough. Machine analysis of bubble chamber pictures. *Proc. Int. Conf. High Energy Accelerators and Instrumentation*, 1959.

[34] C.D. Hu, Y. Chinenov, and T.K. Kerppola. Visualization of interactions among bzip and rel family proteins in living cells using bimolecular fluorescence complementation. *Molecular Cell*, 9:789–798, 2002.

[35] J. Ihmels, S. Bergmann, and N. Barkai. Defining transcription modules using large scale gene expression data. *Bioinformatics*, 20:1993–2003, 2004.

[36] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31:370–377, 2002.

[37] V.R. Iyer, C.E. Horak, C.S. Scafe, D. Botstein, M. Snyder, and P.O. Brown. Genomic binding sites of the yeast cell-cycle transcriptiona factors sbf and mbf. *Nature*, 409:533–538, 2001.

[38] L. Jacob and J.P. Vert. Proteincligand interaction prediction: An improved chemogenomics approach. *Bioinformatics*, 24:2149–2156, 2008.

[39] G. Jones, P. Willett, R. C. Glen, and et al. Development and validation for a genetic algorithm for flexible docking. *J. Mol. Biol.*, 267:727–748, 1997.

[40] H. De Jong, J.L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bull. Math. Biol.*, 66:301–340, 2004.

[41] M. Kanehisa. The KEGG database. *Novartis Foundation Symposium*, 247:91–101, 2002.

[42] M. Kanehisa, S. Goto, M. Hattori, and et al. From genomics to chemical genomics: New developments in kegg. *Nucleic Acids Res.*, 34(DB Issue):D354–D357, 2006.

[43] S. Kauffman. The origins of order: Self-organization and selection in evolution. *Oxford University Press*, 1993.

[44] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Genetic networks with canalyzing boolean rules are always stable. *Procl Natl. Acad. Sci. U.S.A*, 101:17102–17107, 2004.

[45] M. J. Keiser, B. L. Roth, B. N. Armbruster, and et al. Relating protein pharmacology by ligand chemistry. *Nat. Biotech.*, 25:197–206, 2007.

[46] I. M. Keseler, J. Collado-Vides, and S. Gama-Castro *et al.* EcoCyc: a comprehensive database resource for *escherichia coli*. *Nucleic Acids Research*, 33:D334–D337, 2005.

[47] T. Klabunde and G. Hessler. Drug design strategies for targeting g protein-coupled receptors. *Chem. Bio. Chem.*, 3:928–944, 2002.

[48] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13:703–716, 2003.

[49] P. Kuksa, Y. Qi, B. Bai, R. Collobert, J. Weston, V. Pavlovic, and X. Ning. Semi-supervised abstraction-augmented string kernel for multi-level bio-relation extraction. *ECML/PKDD*, 2:128–144, 2010.

[50] F. G. Kuruvilla1, A. F. Shamji, S. M. Sternson, and et al. Dissecting glucose signalling with diversity-oriented synthesis and small-molecule microarrays. *Nature*, 416:653–657, 2002.

[51] G. Lanckriet, M. Cristianini, P. Bartlett, and L.E. Ghaoui. Learning the kernel matrix with semi-denite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[52] F. Li and Y. Yang. Recovering genetic regulatory networks from micro-array data and location analysis data. *Genome Informatics*, 15:131–140, 2004.

[53] G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu. QUBIC: A qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research*, 37:e101, 2009.

[54] X. Liu and L. Wang. Computing the maximum similarity bi-clusters of gene expression data. *Bioinformatics*, 23:50–56, 2006.

[55] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *Journal of Computational Biology and Bioinformatics*, 1:24–45, 2004.

[56] J. Meng and Y. Huang. Biclustering of time series microarray data. *Methods Mol Biol*, 802:87–100, 2012.

[57] C.A. Micchelli and M. Pontil. Learning the kernel function via regularization. *J. Mach. Learn. Res.*, 6:1099–1125, 2006.

[58] B. Mirkin. Mathematical classification and clustering. *Kluwer Academic Publishers*, 1996.

[59] G. M. Morris1, D. S. Goodsell1, R. S. Halliday, and et al. Automated docking using a lamarckian genetic algorithm and empirical binding free energy function. *J. Comput. Chem.*, 19:1639–1662, 1998.

[60] N. Nagamine and Y. Sakakibara. Statistical prediction of proteincchemical interactions based on chemical structure and mass spectrometry data. *Bioinformatics*, 23:2004–2012, 2007.

[61] A. Ng. Feature selection, l1 vs l2 regularization, and rotational invariance. *International Conf. on Mach. Learn. (ICML)*, 69:78, 2004.

[62] S. Nikolajewa, M. Friedel, and T. Wilhelm. Boolean networks with biologically relevant rules show ordered behavior. *Biosystems.*, 90:40–47, 2007.

[63] Y. Ofran, V. Mysore, and B. Rost. Prediction of dna-binding residues from sequence. *Bioinformatics*, 23(13):i347–i353, 2007.

[64] K. Palczewski, T. Kumasaka, T. Hori, and et al. Crystal structure of rhodopsn: A g protein-coupled receptor. *Science*, 289:739–745, 2000.

[65] U. Paul, V. Kaufman, and B. Drossel. Properties of attractors of canalyzing random boolean networks. *Phys. Rev. E. Stat. Nonlin. Soft. Matter. Phys.*, 73:026118, 2006.

[66] A. Prelić, S. Bleuler, P. Zimmermann, and A. Wille. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22:1122–1129, 2006.

[67] M. Rarey, B. Kramer, T. Lengauer, and et al. A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.*, 261:470–489, 1996.

[68] A. Ruepp, A. Zollner, and D. Maier *et al.* The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32:5539–5545, 2004.

[69] I. Schomburg, A. Chang, C. Ebeling, and et al. Brenda, the enzyme database: Updates and major new developments. *Nucleic Acids Res.*, 32:D431–D433, 2004.

[70] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module netwokrs. *J. Mach. Learn. Res.*, 6:557–588, 2005.

[71] D. Shen and Z. Lu. Computation of correlation coefficient and its confidence interval in SAS. http://www2.sas.com/proceedings/sugi31/170-31.pdf.

[72] Y. Shi, Z. Cai, G. Lin, and D. Schuurmans. Linear coherent bi-cluster discovery via line detection and sample majority voting. In *International Conference on Combinatorial Optimization and Applications*, pages 73–84, 2009.

[73] Y. Shi, Y. Guo, and D. Schuurmans. Kernel-based gene regulatory network inference. *International Conference on Computational Systems Bioinformatics*, pages 156–165, 2010.

[74] Y. Shi, M. Hasan, Z. Cai, G. Lin, and D. Schuurmans. Linear coherent bicluster discovery via beam detection and sample set clustering. In *International Conference on Combinatorial Optimization and Applications*, page In progress, 2010.

[75] B. K. Shoichet, D. L. Bodian, and I. D. Kuntz. Molecular docking using shape descriptors. *J. Comput. Chem.*, 13:380–397, 1992.

[76] I. Simon, J. Barnett, N. Hannett, C.T. Harbison, N.J. Rinaldi, T.L. Volkert, J.J. Wyrick, J. Zeitlinger, D.K. Gifford, T.S. Jaakkola, and R.A. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, 2001.

[77] P. Simon, L. Kevin, and T. James. Grafting: Fast, incremental feature selection by gradient descent in function space. *J. Mach. Learn. Res.*, 3:1333–1356, 2003.

[78] T.F. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Biol*, 147:195–197, 1981.

[79] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[80] E. Van Someren, L. Wessels, and M. Reinders. Learning modeling of genetic networks from experimental data. *Intelligient Systems for Molecular Biology (ISMB)*, pages 355–366, 2000.

[81] O. D. Sparkman. Mass spectrometry desk reference. *Pittsburgh: Global View Pub*, ISBN 0-9660813-2-3, 2000.

[82] P. Tamayo, D. Slonim, and J. Mesirov *et al.* Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *PNAS*, 96:2907–2912, 1999.

[83] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:136–144, 2002.

[84] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[85] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[86] S. Wang. Reconstructing genetic networks from time ordered gene expression data using bayesian method with global search algorithm. *J. Bioinform. Comput. Biol.*, 2:441–458, 2004.

[87] P. H. Westfall and S. S. Young. Resampling-based multiple testing. *Wiley, New York*, 1993.

[88] D. L. Wheeler, T. Barrett, D. A. Benson, and et al. Database resources of the national center for biotechnology information. *Nucleic Acids Res.*, 34:D173–D180, 2006.

[89] D. Wishart, C. Knox, A. Guo, and et al. Drugbank: A knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Res.*, 36(sup1):D901–6, 2007.

[90] X. Zhang G.Lin D. Schuurmans Y. Shi, X. Liao. Sparse learning based linear coherent bi-clustering. *WABI. Lecture Notes in Bioinformatics*, 7534:346–364, 2012.

[91] Y. Yamanishi, M Araki, A. Gutteridge, and W. Honda. Prediction of drugctarget interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24:232–240, 2008.

[92] Yoshihiro Yamanishi, Masaaki Kotera, Minoru Kanehisa, and Susumu Goto. Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, 26:i246–i254, 2010.

[93] M.A. Yildirim, K.I Goh, M.E Cusick, A.L. Barabsi, and M. Vidal. Drugctarget network. *Nature Biotechnology*, 25:1119–1126, 2007.

[94] J. Yu, V. Smith, P. Wang, A. Hartemink, and E. Jarvis. Advances to bayesian network inference for generating causual networks from observational biological data. *Bioinformatics*, 20:3594–3603, 2004.

[95] W. Zhong and J. Kwok. Efficient sparse modeling with automatic feature grouping. *ICML*, 2011.

[96] X. Zhou and Z. Su. EasyGO: Gene ontology-based annotation and functional enrichment analysis tool for agronomical species. *BMC Genomics*, 8:246, 2007.

[97] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. *Neural Information Processing Systems*, page 16, 2003.

[98] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005.

[99] M. Zou and S. Conzen. A new dynamic bayesian network (dbn) approach for identifying gene regulatory netwokrs from time course microarray data. *Bioinformatics*, 21:71–79, 2005.

Appendix A

Line Detection based Bi-clustering

A.1 The LCBD Algorithm

Let $A(I, J)$ be an $n \times m$ real valued matrix, where $I = \{1, 2, 3, ..., n\}$ is the set of genes and $J = \{1, 2, 3, ..., m\}$ is the set of samples. The element a_{ij} of $A(I, J)$ represents the expression level of gene i under sample j. A row vector $A(i, J)$ and a column vector $A(I, j)$ represents the ith gene over all the samples and the jth sample over all the genes, respectively. Our algorithm is composed of three major steps.

In the first step, for each pair of genes $A(p, J)$ and $A(q, J)$, where $p, q \in \{1, 2, 3, ...n\}$ and $p \neq q$, I construct a two-dimension binary matrix that represents the 2D image of the two vectors with x-coordinates $A(p, J)$ and y-coordinates $A(q, J)$, respectively. A pixel in the 2D image is denoted by a 1 in the binary matrix. Using the binary matrix as input, I then identify lines in the 2D image based on the Hough transform. The Hough transform technique works on the following principle: First note that each point (pixel) in a 2D image can be passed through by an infinite number of lines, and each of which can be parameterized by r and θ, where r is the perpendicular distance between the origin and the line and θ is the angle between the perpendicular line and the x-coordinate. Then note that the set of lines that pass through a point forms a sinusoidal curve in the $r - \theta$ coordinate space. Now, if there is a common line that passes through a set of points in the original 2D image, their corresponding sinusoidal curves must have a point of intersection in the $r - \theta$ space. So by finding a point of intersection in $r - \theta$ space,

one can identify a line that passes through a set of points in the original 2D space. Each line in the 2D image is a linear correlation between a pair of genes under a subset of samples. To allow for possible overlaps in the final bi-clusters, I let the Hough transform identify at most k non-reduplicative lines. Therefore, for each pair of genes, one can collect at most k sample sets over which the two genes are linearly correlated. After collecting sample sets for each gene pair, an $n \times n$ upper trianglular matrix can be obtained, where each element contains at most k sample sets (see Figure A.1 for illustration). Denote each element in the matrix as S_{ij}. Note that for each 2D image, the horizontal lines and vertical lines in the 2D image are not eliminated since they might not represent linear correlations.

In the second step, for vector of sample sets, S_{iJ}, I count the samples that appear in each element of S_{iJ}. I then collect the top w voted samples into a sample pool and the corresponding genes who voted for these samples into a gene pool. The sample and gene pools thus constitute an initial bi-cluster. Then, for the remaining samples, I iteratively add them and their corresponding gene into the sample and gene pools, respectively, as long as by adding them the mean gene-gene correlation coefficient of the current bi-cluster remains above a threshold. The user specified parameter w should be greater than 3, because one can always draw a line between any 2 random points but the possibility that more than 2 random points lie on the same line is very small unless there is a linear correlation. In this step, each sample sets vector S_{iJ} will construct at most one bi-cluster that necessarily contains gene i.

In the third step, I remove redundancy in the bi-cluster sets generated in step two. If two bi-clusters share more than 60% identical elements, one of the two will be removed depending on which has more identical elements. Algorithm 1 describes the LCBD algorithm.

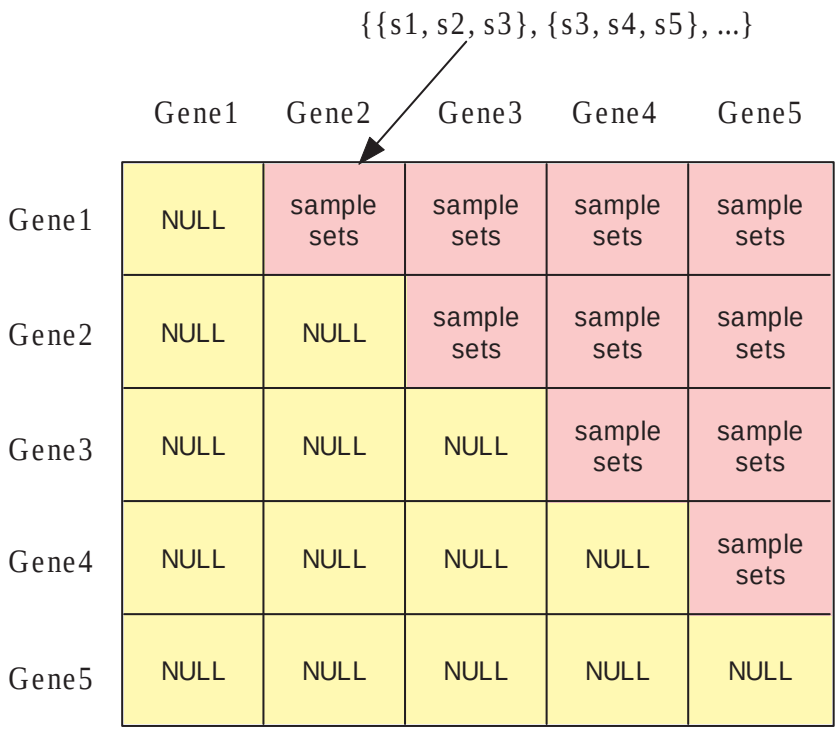


Figure A.1: Illustration of an $n \times n$ gene pairwise sample sets matrix

Algorithm 1 The LCBD Algorithm

Input An $n \times m$ real value matrix $A(I, J)$, k, w.
Output A set of bi-clusters $A(g_i, s_i)$, where $g_i \subseteq I$ and $s_i \subseteq J$.

for i = 1 to n **do**
  for $j = i + 1$ to n **do**
    Construct binary matrix $B_{i,j}$ for vectors $A(i, J)$ and $A(j, J)$;
    Do Hough transform based on $B_{i,j}$ and k to obtain a set of sample sets, S_{iJ};
  end for
end for

for i = 1 to n **do**
  Select the top w most voted samples in S_{iJ} as the initial sample pool s_i;
  Select the genes whose corresponding gene pair sample sets contain all the initial samples g_i;
  Construct the initial bi-cluster $A(g_i, s_i)$;
  while gene-wise mean correlation coefficient of $A(g_i, s_i) <$ threshold **do**
    Add the most voted sample in the leftover sample sets to the sample pool s_i;
    Add the corresponding gene into the gene pool g_i;
    Update bi-cluster $A(g_i, s_i)$;
  end while
end for

Remove redundant bi-clusters in the set $A(g_i, s_i)$ that has $> 60\%$ overlapping elements.
Output the set of bi-clusters $A(g_i, s_i)$;

A.2 Results

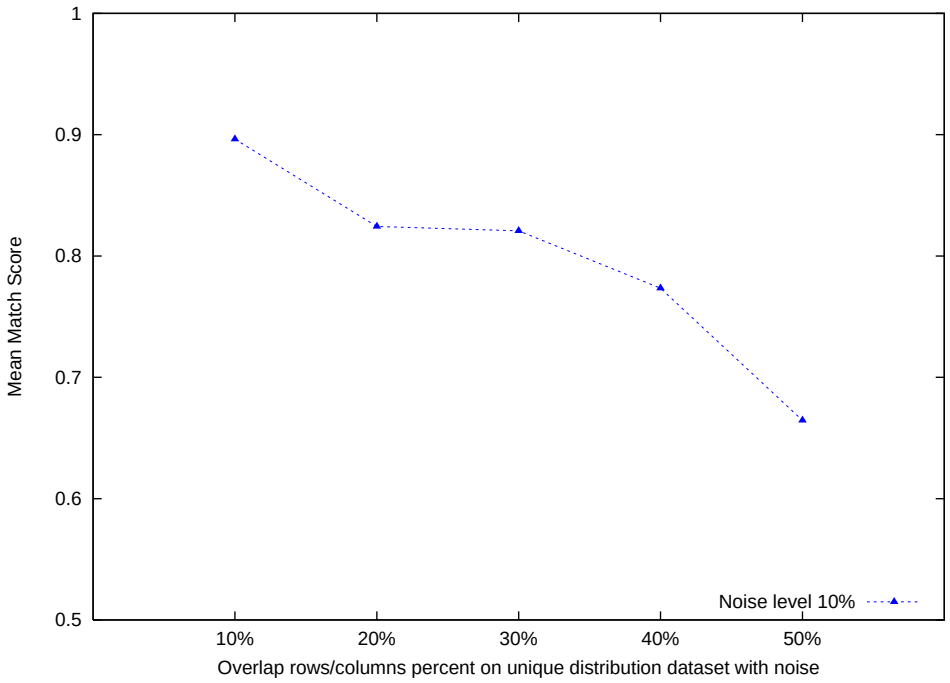


Figure A.2: Match score of the LCBD method of different bi-cluster overlapping rate on synthetic dataset under unique distribution

Appendix B

Beam Detection based Bi-clustering

B.1 The LinCoh Algorithm

Let $A(G, S)$ be an $n \times m$ gene expression data matrix, where $G = \{1, 2, \ldots, n\}$ is the set of gene indices and $S = \{1, 2, \ldots, m\}$ is the set of sample (condition) indices. Its element e_{ij} is the expression level of gene i in sample j. Our LinCoh algorithm consists of two major steps, described in the next two subsections.

Step one: establishing pairwise gene relations. For each pair of genes $p, q \in \{1, 2, \ldots, n\}$, I plot their expression levels in all samples in a 2D plane, as shown in Figure 2.6, where a point (x, y) represents a sample in which gene p has expression level x and gene q has expression level y. The goal of this step is to detect a correlation between every pair of genes on a subset of samples, if any. Such a subset of samples must evidence the correlation, in the way that the two genes are co-up-regulated (or co-down-regulated) in these samples [53]. I define a *beam* $B_{\theta,\beta,\gamma}$ in the 2D plane to be the set of points on the plane that are within distance $\frac{1}{2}\beta$ to a straight line that depends on θ and γ. Here θ is the *beam angle*, β is the *beam width*, and γ is the *beam offset*. They are all search parameters, but I am able to pre-determine some best values or ranges of values for them.

Let μ_p and σ_p (μ_q and σ_q) denote the mean expression level of gene p (q, respectively) across all samples and the standard deviation. To identify the subset of supporting samples for this gene pair, $S_{\theta,\beta,\gamma} = S \cap B_{\theta,\beta,\gamma}$, I seek for a beam $B_{\theta,\beta,\gamma}$ in the 2D plane that aligns approximately the main diagonal (or the antidiagonal) of the rectangle defined by $\{(\mu_p - \sigma_p, \mu_q - \sigma_q), (\mu_p + \sigma_p, \mu_q - \sigma_q), (\mu_p + \sigma_p, \mu_q +$

$\sigma_q), (\mu_p - \sigma_p, \mu_q + \sigma_q)\}$. Such an approximate alignment optimizes the following objective function, which robustly leads to good quality bi-clusters:

$$\begin{aligned} & \max_{\theta,\beta,\gamma} \; W_{S_{\theta,\beta,\gamma}} \cdot D_{S_{\theta,\beta,\gamma}} \\ \text{subject to:} \quad & \Big|corr\big(A(p, S_{\theta,\beta,\gamma}), A(q, S_{\theta,\beta,\gamma})\big)\Big| \geq t_{cc}. \end{aligned}$$

In the above maximization problem, $D_{S_{\theta,\beta,\gamma}}$ is the vector of the Euclidean distances of the samples inside the beam, *i.e.* $S_{\theta,\beta,\gamma}$, to the line passing through (μ_p, μ_q) and perpendicular to (called the *midsplit line* of) the beam center line; $W_{S_{\theta,\beta,\gamma}}$ is a weight vector over the samples in $S_{\theta,\beta,\gamma}$, and I use Shepard's function $w_j = d_j^r / \sum d_j^r$ with parameter $r \geq 0$ to weight sample $j \in S_{\theta,\beta,\gamma}$ (to weight more on distant samples but less on samples nearby the midsplit line); In the constraint, $|corr(\cdot,\cdot)|$ is the absolute correlation coefficient between the two genes p and q, calculated over only the samples in $S_{\theta,\beta,\gamma}$; t_{cc} is a pre-determined correlation threshold.

The output of the maximization problem is $S_{\theta,\beta,\gamma}$, which is either empty, indicating that no meaningful relationship between the two genes was found, or otherwise a subset of samples that evidence a meaningful correlation between genes p and q. According to our extensive preliminary experiments, the bi-clustering results are of high quality when the correlation threshold t_{cc} is larger than 0.75; and it is set to 0.90 and 0.75, respectively, on synthetic datasets and real datasets.

I implemented a heuristic process to search for the beam, whose center line is initialized to be the line passing through the main diagonal (for positive correlation) or the antidiagonal (for negative correlation) of the rectangle in the expression plot. The beam width β is fixed at a certain portion of $4\sigma_p\sigma_q/\sqrt{{\sigma_p}^2 + {\sigma_q}^2}$; again supported by the preliminary experiments, a constant portion in between 0.8 and 1.0 is sufficient to capture most meaningful correlations; and I fix it at 0.8 for both synthetic datasets and real datasets. That is, $\beta = 0.8 \times 4\sigma_p\sigma_q/\sqrt{{\sigma_p}^2 + {\sigma_q}^2}$. To determine the beam angle θ in the positive correlation case, I define the search axis to be the midsplit line of the main diagonal; a small interval is placed on the search axis centering at (μ_p, μ_q), which is for a pivot point to float within; around each position of the pivot point, whose distance to the center point (μ_p, μ_q) is denoted as γ, different angles (the θ) are searched over to find a beam center line; each re-

sultant beam is tested for the constraint satisfaction in the maximization problem, and discarded otherwise; among all those beams that satisfy the constraint, the one maximizing the objective function is returned as the target beam.

For evaluating the objective function, I have tested multiple values of r and found that 0 gives the most robust bi-clustering results. Therefore, r is set to 0 as default. For each sample j inside the beam, its distance d_j to the mid-split line of the beam center line is rounded to 0 or 1 using a threshold of $\sqrt{{\sigma_p}^2 + {\sigma_q}^2}$. When the target beam is identified, though might not be the true optimum to the objective function, the sample set $S_{\theta,\beta,\gamma}$ is further partitioned into *outer* sample set (containing the samples with distance d_j rounded to 1) and *inner* sample set (containing the rest). Gene pairs, together with non-empty outer and inner sample sets, are sent to Step two for clustering.

Step two: sample set clustering. Step one generates an outer sample set and an inner sample set for each gene pair. In this step, two $n \times n$ matrices are constructed: in the *outer matrix* M^o, the element m^o_{pq} is the outer sample set for gene pair p and q; likewise, in the *inner matrix* M^i, the element m^i_{pq} is the inner sample set for gene pair p and q. I next process these two matrices to robustly find bi-clusters.

First, I want to filter out small outer sample sets that indicate insignificant correlations for gene pairs. To this purpose, I select roughly the largest 0.15% outer sample sets among all for clustering, which are of 99.7% confidence.This confidence level is set after testing on a randomly generated datasets, with 68%, 95%, 99.7% confidence levels according to the 68-95-99.7 rule. Observing that two disjoint gene pairs could have the same outer sample sets but very different expression patterns, simply using outer sample sets to construct bi-clusters might lead to meaningless bi-clusters. In our LinCoh algorithm, genes are used as bridges to group similar outer sample sets to form bi-clusters, since linear correlation is transitive. I first define the similarity between two outer sample sets m^o_{pq} and m^o_{rs} as $\mathrm{sim}(m^o_{pq}, m^o_{rs}) = |m^o_{pq} \cap m^o_{rs}|/|m^o_{pq} \cup m^o_{rs}|$, which is a very popular measure in the literature. Next, I rank genes in the descending order of the number of associated non-empty outer sample sets. Iteratively, the gene at the head of this list is used as the *seed* gene, to collect all its associated (non-empty) outer sample sets. These

outer sample sets are partitioned into clusters using a density based clustering algorithm similar to DBSCAN [20], and the densest cluster is returned, which is defined as a cluster whose central point has the most close neighbors. An initial bi-cluster is formed on the union S_1 of the outer sample sets in the densest cluster, and the set G_1 of the involved genes.

The quality of the bi-cluster (G_1, S_1) is evaluated by its *average absolute correlation coefficient*,

$$\text{aacc}(G_1, S_1) = \frac{\sum_{p,q \in G_1} |corr(A(p, S_1), A(q, S_1))|}{(|G_1|^2 - |G_1|)}. \tag{B.1}$$

The initial bi-cluster (G_1, S_1) is refined in three steps to locally improve its quality. In the first step, all samples in S_1 are sorted in decreasing frequency of occurrence in all the outer sample sets of the seed gene; the lowest ranked sample is removed if this removal improves the quality of the bi-cluster, or otherwise the first step is done. Secondly, every gene in G_1 is checked to see whether its removal improves the quality of the bi-cluster, and if so it is removed from G_1. At the end, if the minimum gene pairwise absolute correlation coefficient of the bi-cluster is smaller than a threshold, the bi-cluster is considered as of low quality and discarded. By examining values from 0.50 to 0.99, our preliminary experiments showed that a high threshold in between 0.7 and 0.9 is able to ensure good quality bi-clusters, and it is set to 0.8 in all our final experiments. In the last step of bi-cluster (G_1, S_1) refinement, the inner sample sets of the gene pairs from G_1 are collected; their samples are sorted in decreasing frequencies in these inner sample sets; using this order, samples are added to S_1 as long as their addition passes the 0.8 minimum pairwise absolute correlation coefficient. A final bi-cluster (G_1, S_1) is thus produced.

Subsequently, all genes from G_1 are removed from the gene list, and the next gene is used as the seed gene for finding the next bi-cluster. The process iterates until the gene list becomes empty. I remark that a gene can participate in multiple bi-clusters, but it serves as a seed gene at most once. At the end, when two bi-clusters overlap more than 60% area, the one of smaller size is treated as redundant and discarded [53]. The pseudocode for the LinCoh algorithm 2 is provided below.

Algorithm 2 The LinCoh Algorithm

Input An $n \times m$ real value matrix $A(I, J)$, T_{close}, T_{minCC}
Output A set of bi-clusters $A(g_i, s_i)$, where $g_i \subseteq I$ and $s_i \subseteq J$.

for i = 1 to n **do**
  for $j = i + 1$ to n **do**
    $M_O(i, j) = NULL$, $M_I(i, j) = NULL$;
    $\theta_{rec} = NULL$, $\beta_{rec} = NULL$, $\gamma_{rec} = NULL$;
    for A set of beam parameters (θ, β, γ) **do**
      if $W_{S_{outer(\theta,\beta,\gamma)}} \cdot D^T_{S_{outer(\theta,\beta,\gamma)}} > |M_O(i, j)|$ **then**
        $M_O(i, j) = S_{outer(\theta,\beta,\gamma)}$;
        $\theta_{rec} = \theta$, $\beta_{rec} = \beta$, $\gamma_{rec} = \gamma$;
      end if
    end for
    $M_I(i, j) = S_{inner(\theta_{rec},\beta_{rec},\gamma_{rec})}$;
  end for
end for

for i = 1 to n **do**
  for $j = i + 1$ to n **do**
    if $M_O(i, j) < \mu_{ss} + \alpha \cdot \sigma_{ss}$ **then**
      $M_O(i, j) = NULL$, $M_I(i, j) = NULL$;
    end if
  end for
end for

for i = 1 to n **do**
  $SS_i = \bigcup_{j \in J}(M_O(i, j))$;
end for
$GeneList_{ss}$ = DescendSort($Genes$) based on $|SS_i| \neq NULL$ of each $i \in I$;
$BiclusterPool = NULL$;
while $GeneList_{ss} \neq EMPTY$ **do**
  $SeedGene = Pop(GeneList_{ss})$;
  Construct similarity matrix $Matrix_{ss}$ for $SS_{seedGene}$ elements based on $M_S(SS_i, SS_j) = \frac{|SS_i \cap SS_j|}{|SS_i \cup SS_j|}$;
  Find the centroid sample set $SS_{centroid}$ that has the most *close* ($M_S(S_i, S_j) \geq T_{close}$) neighbors, $SS_{neighbors}$;
  $GenePool = \bigcup(G_i \in G_{SS_{centroid}} \bigcup G_{SS_{neighbors}})$;
  $SamplePool = \bigcup(S_j \in SS_{centroid} \bigcup SS_{neighbors})$;
  $BiCluster_{initial} = A(GenePool, SamplePool)$;
  $BiCluster_{refined} = RefineBicluster(BiCluster_{initial})$
  if MinAbsCC($BiCluster_{refined}$) $\geq T_{minCC}$ **then**
    BiclusterPool.add($BiCluster_{refined}$);
  end if
end while
$Biclusters_{final} = RedundantRemoval(BiclusterPool)$

B.2 Results

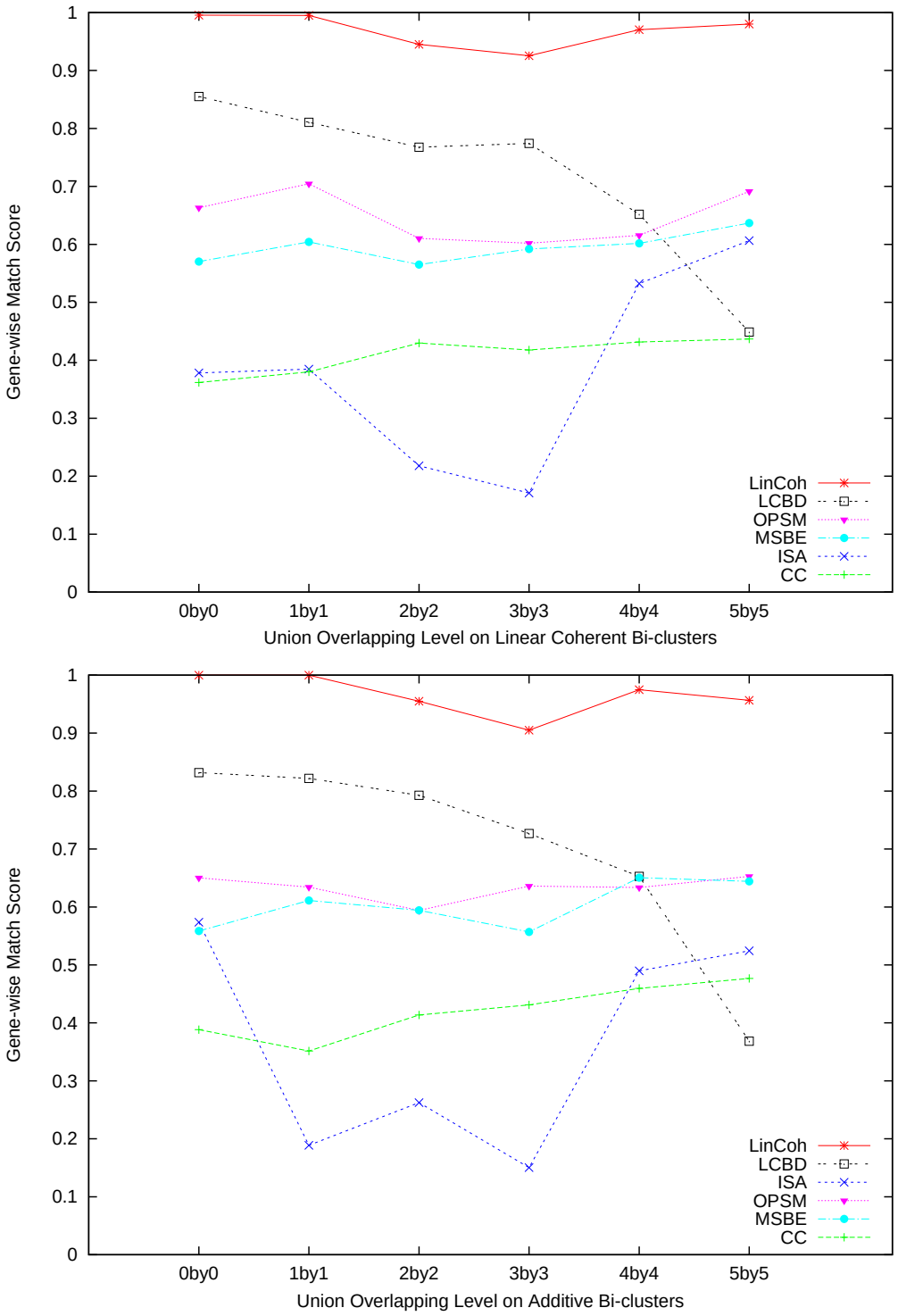


Figure B.1: The gene match scores of the five algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the union overlap model.

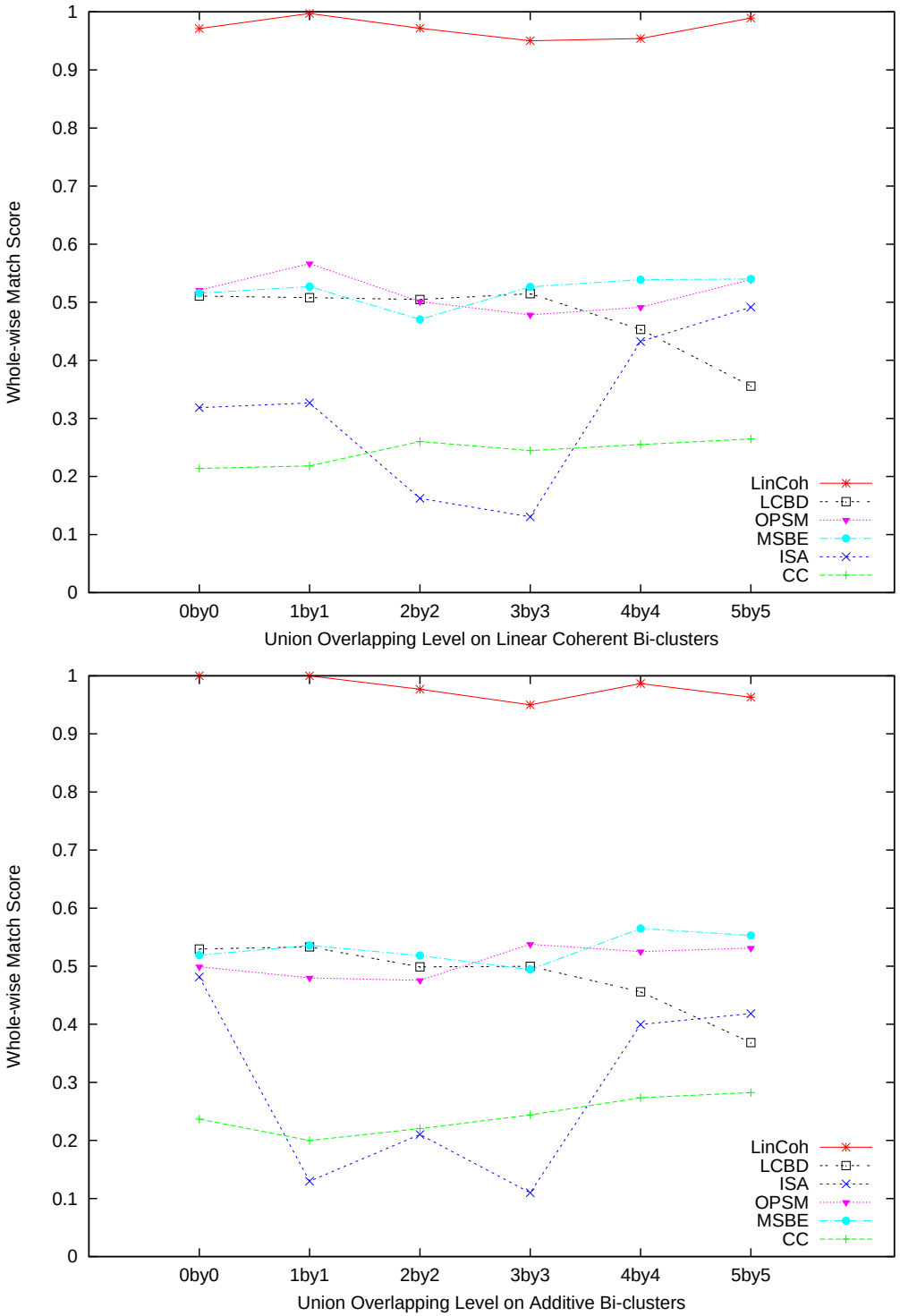


Figure B.2: The overall match scores of the five algorithms for recovering linear coherent and additive bi-clusters, under six different amounts of overlap using the union overlap model.

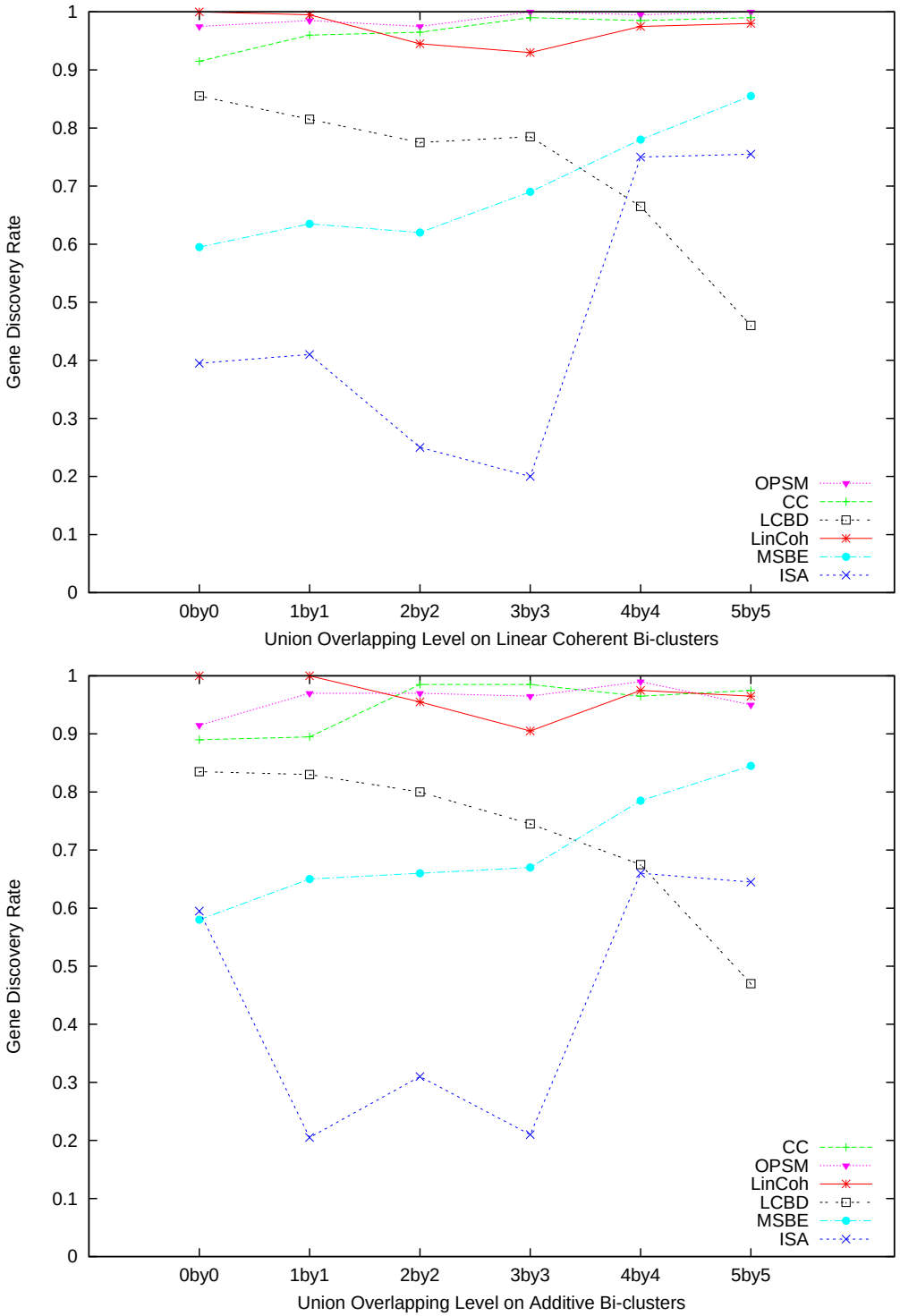


Figure B.3: The gene discovery rates of the five algorithms for recovering linear coherent and additive bi-clusters, under six different amounts of overlap using the union overlap model.

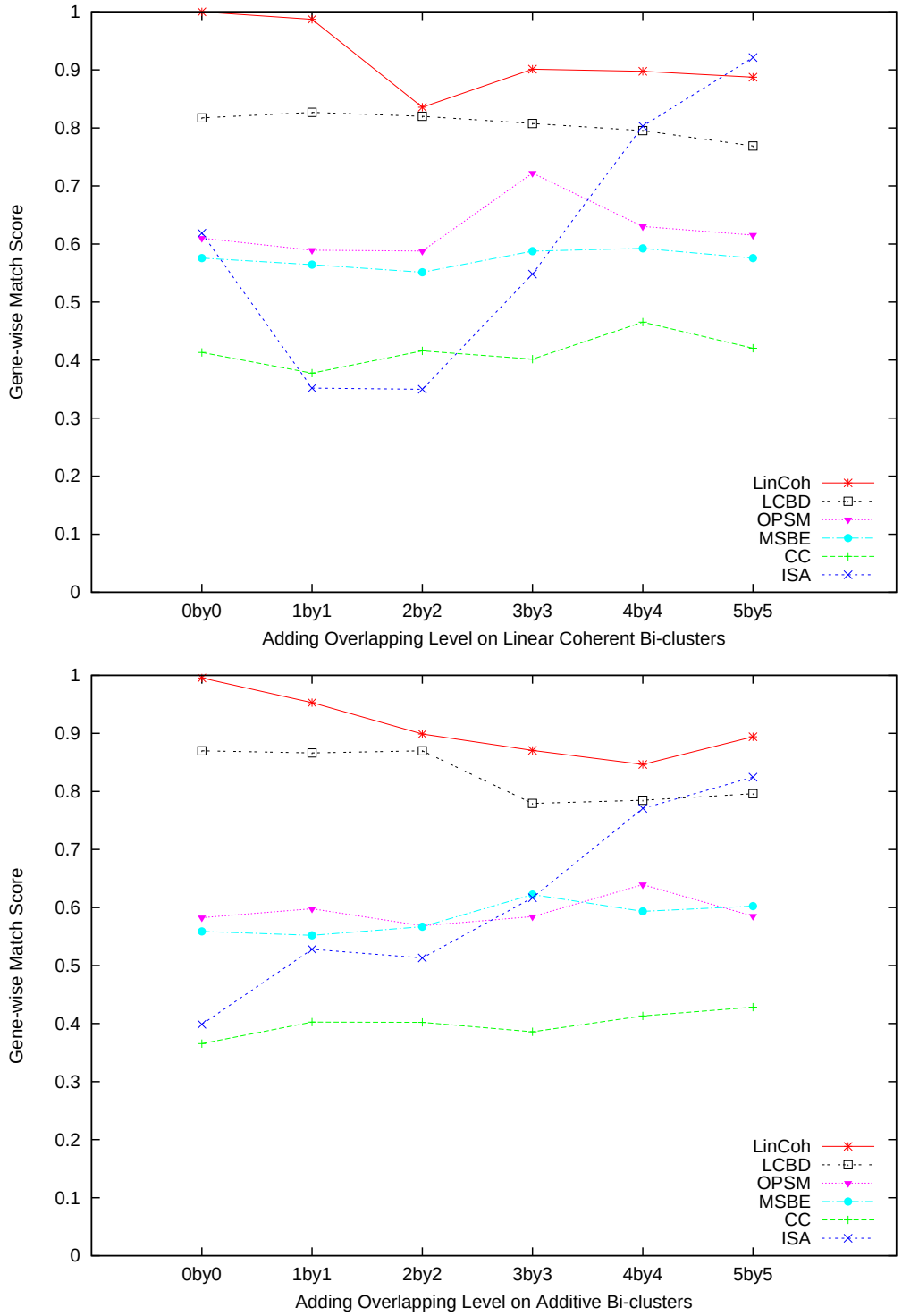


Figure B.4: The gene match scores of the five algorithms for recovering linear coherent and additive bi-clusters, under six different amounts of overlap using the additive overlap model.

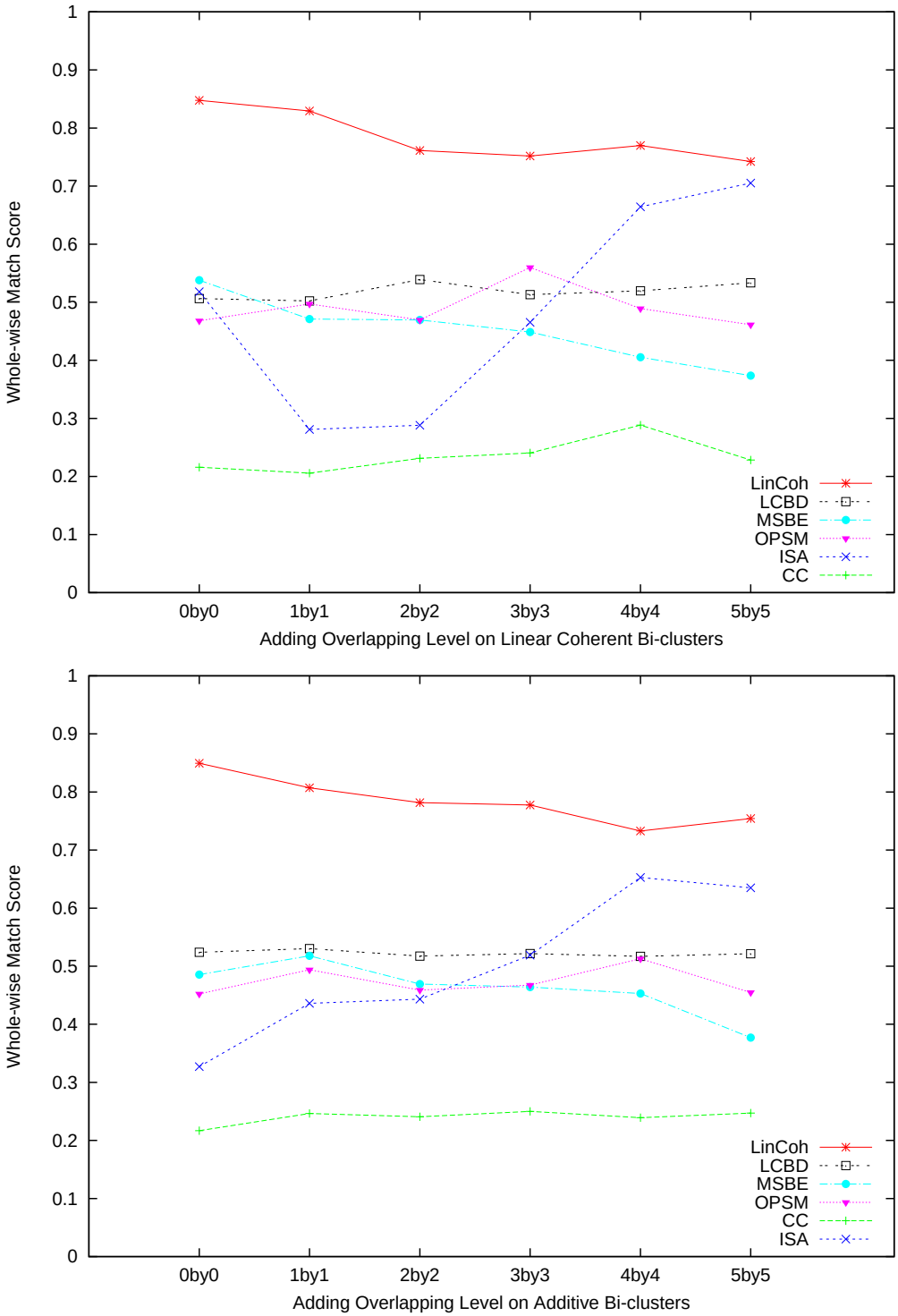


Figure B.5: The overall match scores of the five algorithms for recovering linear coherent and additive bi-clusters, under six different amounts of overlap using the additive overlap model.

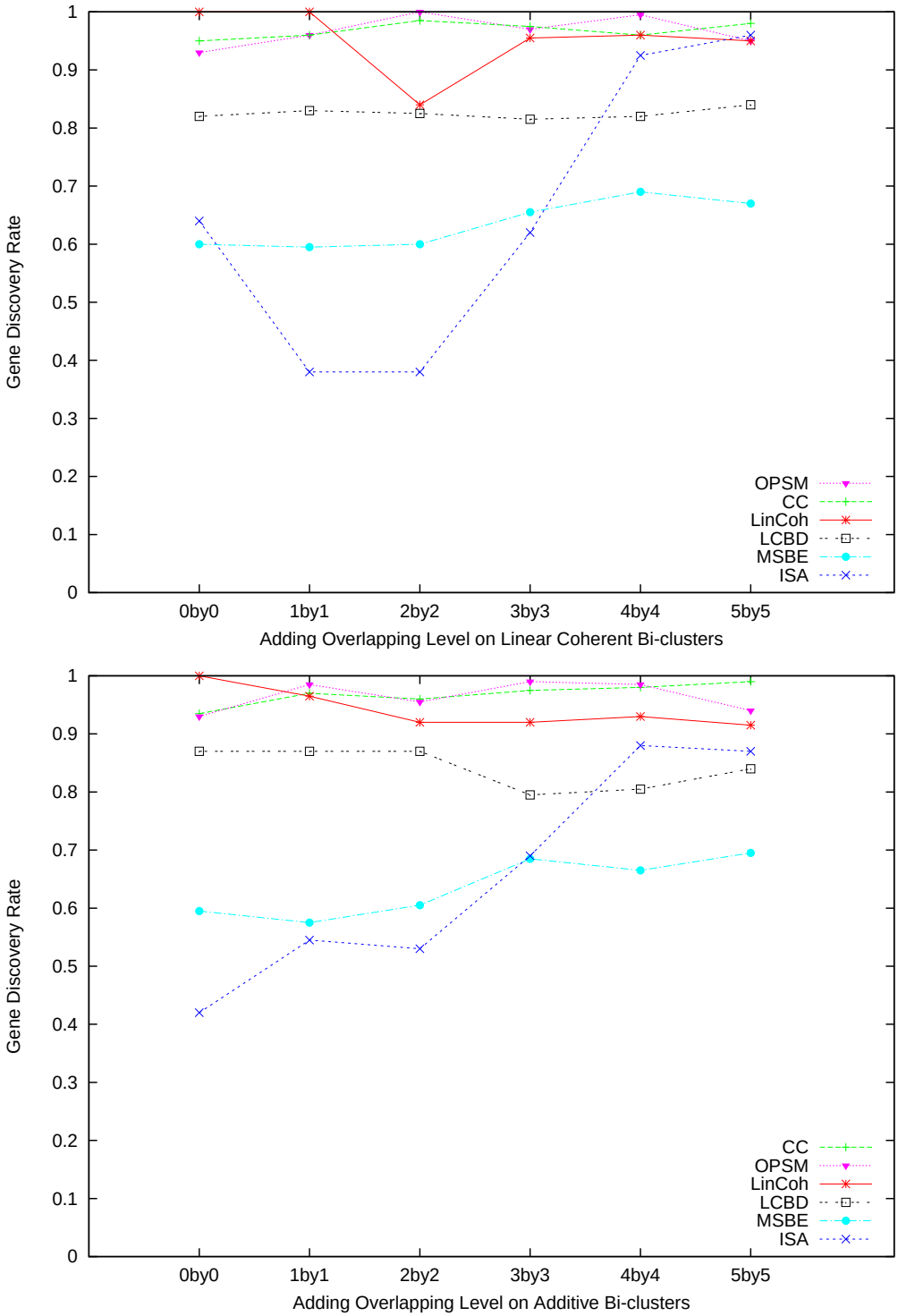


Figure B.6: The gene discovery rates of the five algorithms for recovering linear coherent and additive bi-clusters, under six different amounts of overlap using the additive overlap model.

Table B.1: Statistics of different algorithms' bi-clustering results and the numbers of functional terms enriched on different databases.

	#Bi-clusters	$\mu_{\|gene\|}$	$\sigma_{\|gene\|}$	$\mu_{\|sample\|}$	$\sigma_{\|sample\|}$	#Terms enriched (GO, KEGG, MIPS/regulons)
(Yeast)						
LinCoh	100	61.84	38.43	133.09	18.09	5, 7, 5
LCBD	132	46.46	17.53	13.35	4.28	10, 6, 11
ISA	47	67	34.54	8.4	1.78	15, 13, 18
OPSM	14	423.29	728.95	9.07	5.14	1, 1, 2
MSBE	40	19.25	8.32	18.68	8.22	8, 4, 6
CC	10	297.7	304.18	60.8	23.46	6, 4, 8
(E.coli)						
LinCoh	100	9.63	7.66	141.43	34.04	24, 24, 22
LCBD	155	485.05	336.63	15.37	22.95	23, 22, 33
ISA	34	124.21	42.18	13.88	6.11	11, 10, 13
OPSM	14	419.29	744.35	8.93	4.8	8, 4, 5
MSBE	9	82.67	18.1	80.22	19.18	1, 3, 4
CC	10	309.9	950.15	31.4	81.74	2, 2, 2

Table B.2: For all the gene pairs with absolute correlation coefficient ≥ 0.8, the number of pairs that have between 0 and 7 common GO terms.

Term count	yeast	E.coli
0	909	2680
1	18860	3485
2	7898	1533
3	1839	490
4	165	239
5	30	52
6	6	18
7	4	0
Overall	28802	5817

Table B.3: The top 10 gene pairs' common GO terms and their counts.

yeast		E.coli	
GO term	Count	GO term	Count
GO:0006412	8353	GO:0006412	811
GO:0000723	1920	GO:0008652	680
GO:0000027	1615	GO:0001539	388
GO:0000028	1070	GO:0006810	317
GO:0006365	969	GO:0006355	234
GO:0006413	893	GO:0006811	195
GO:0030488	782	GO:0006865	183
GO:0006364	720	GO:0006260	127
GO:0030490	683	GO:0046677	115
GO:0006360	424	GO:0008152	111

Appendix C

Sparse Learning based Bi-clustering

C.1 The SLLB Algorithm

Algorithm 3 The SLLB Algorithm

Input: M, β_1, β_2, ϵ.
Output: a set of linear coherent bi-clusters.

$A = \mathbf{1} \cdot \mathbf{1}^T$
$b_{ij} = \bar{m}_{i:} - a_{ij}\bar{m}_{j:}$ for $i, j \in [1, 2, ...n]$.
$s_{ik} = 1$ if $d'_{ijk} <= e^{d''_{ijk}} - 1$, $s_{ik} = 0$ otherwise, for $i \in [1, 2, ...n]$, $k \in [1, 2, ...p]$.
while $\Delta L > \epsilon$ **do**
  for each i, j **do**
    if $\sum_k \frac{s_{ik}}{d_{ijk}}(m_{ik} - a_{ij}m_{jk} - b_{ij})^2 < \beta_2$ **then**
      $w_{ij} = 1$
    else
      $w_{ij} = 0$
    end if
  end for
  for each i, j **do**
    if $\sum_j \frac{w_{ij}}{d_{ijk}}(m_{ik} - a_{ij}m_{jk} - b_{ij})^2 < \beta_1$ **then**
      $s_{ik} = 1$
    else
      $s_{ik} = 0$
    end if
  end for
  for each i, j **do**
    $(a_{ij}, b_{ij})^T = (X_{ij}^T \triangle(\mathbf{s_{i:}} \bullet \mathbf{d_{ij:}})X_{ij})^{-1} X_{ij}^T \triangle(\mathbf{s_{i:}} \bullet \mathbf{d_{ij:}})\mathbf{y_{ij}}$
  end for
  Calculate loss change ΔL.
end while
Construct bi-clusters from W, S and remove redundant bi-clusters.

C.2 Results

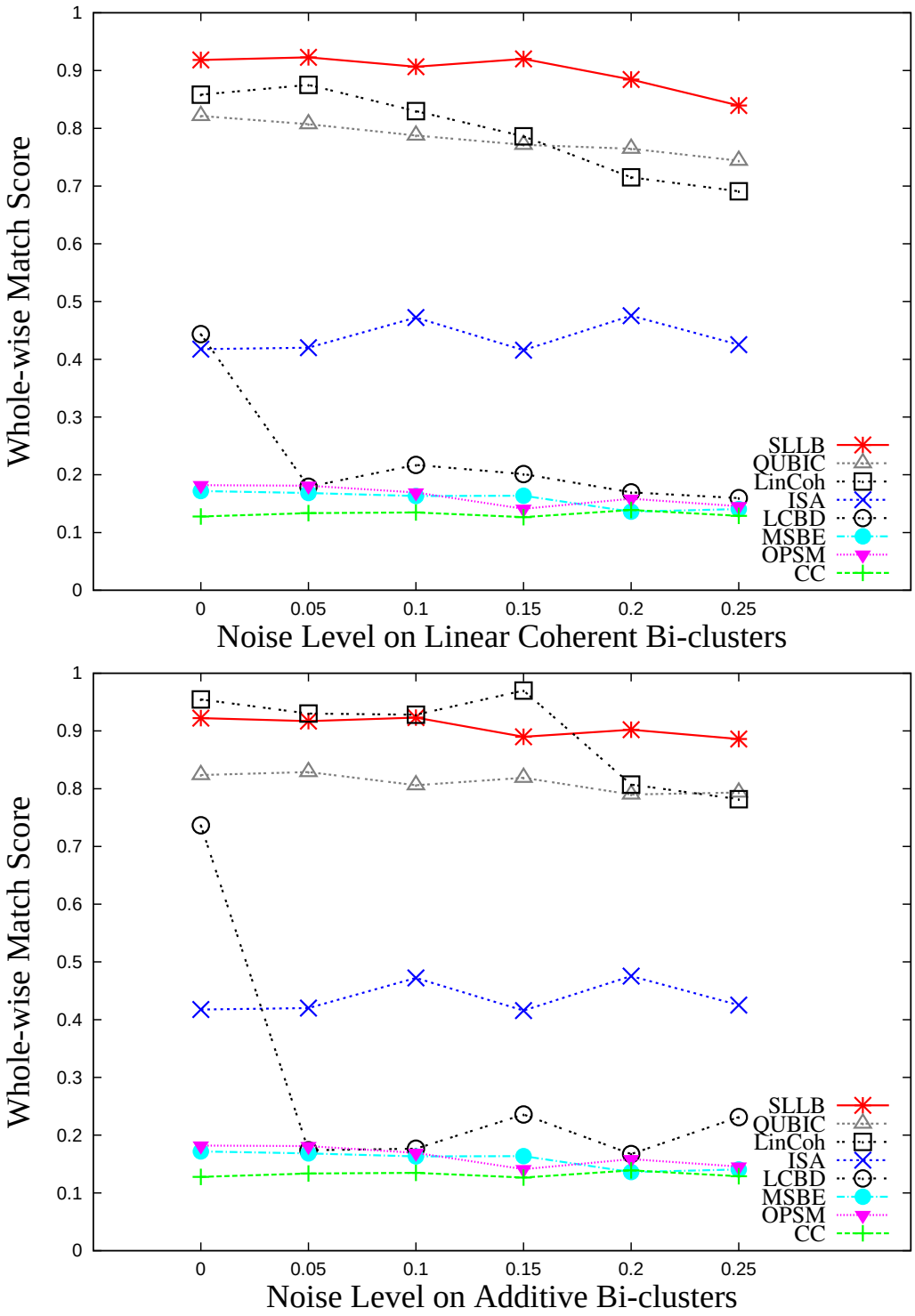


Figure C.1: The overall match scores of the eight algorithms for recovering linear coherent and additive bi-clusters, at six different noise levels.

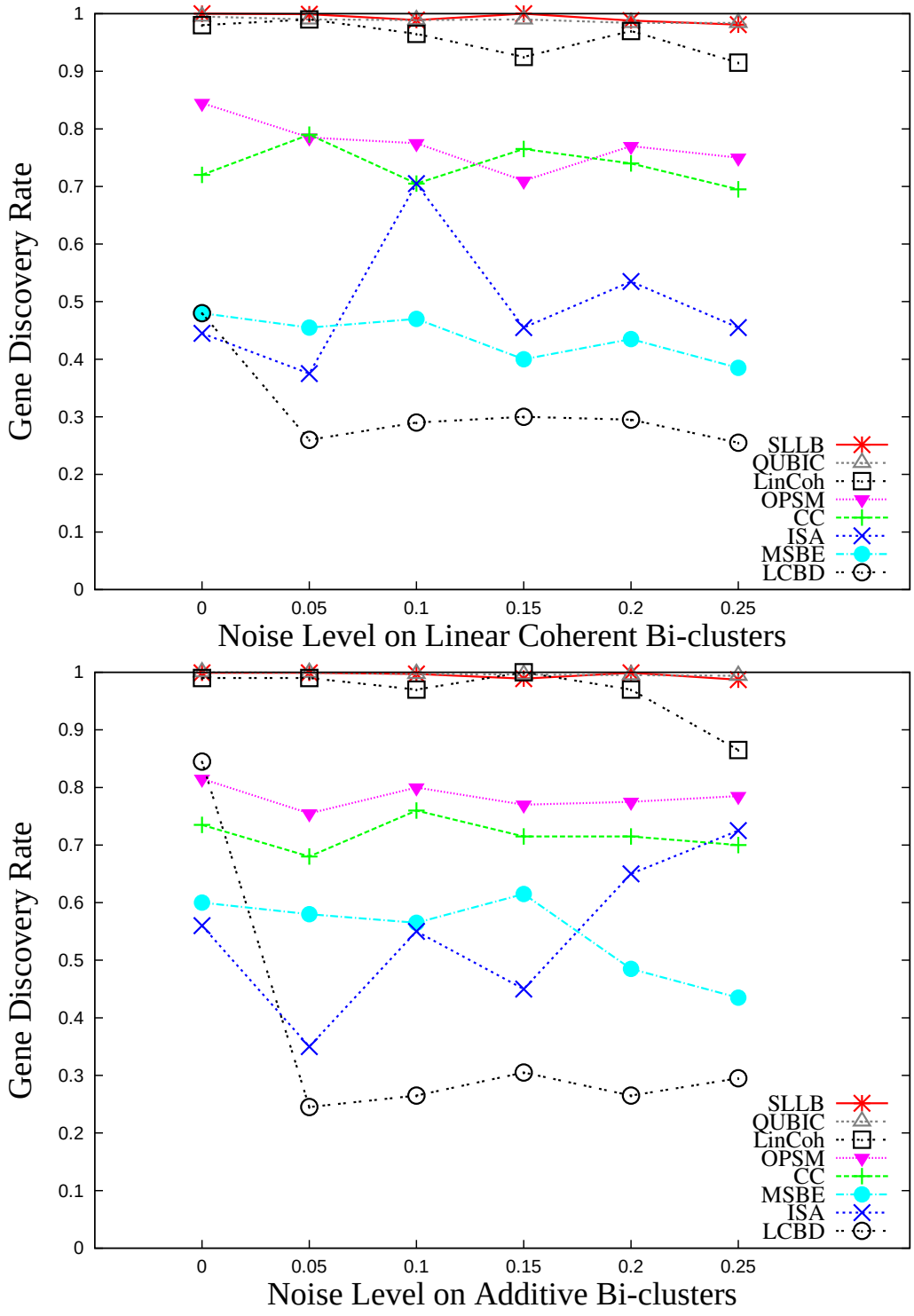


Figure C.2: The gene discovery rates of the eight algorithms for recovering linear coherent and additive bi-clusters, at six different noise levels.

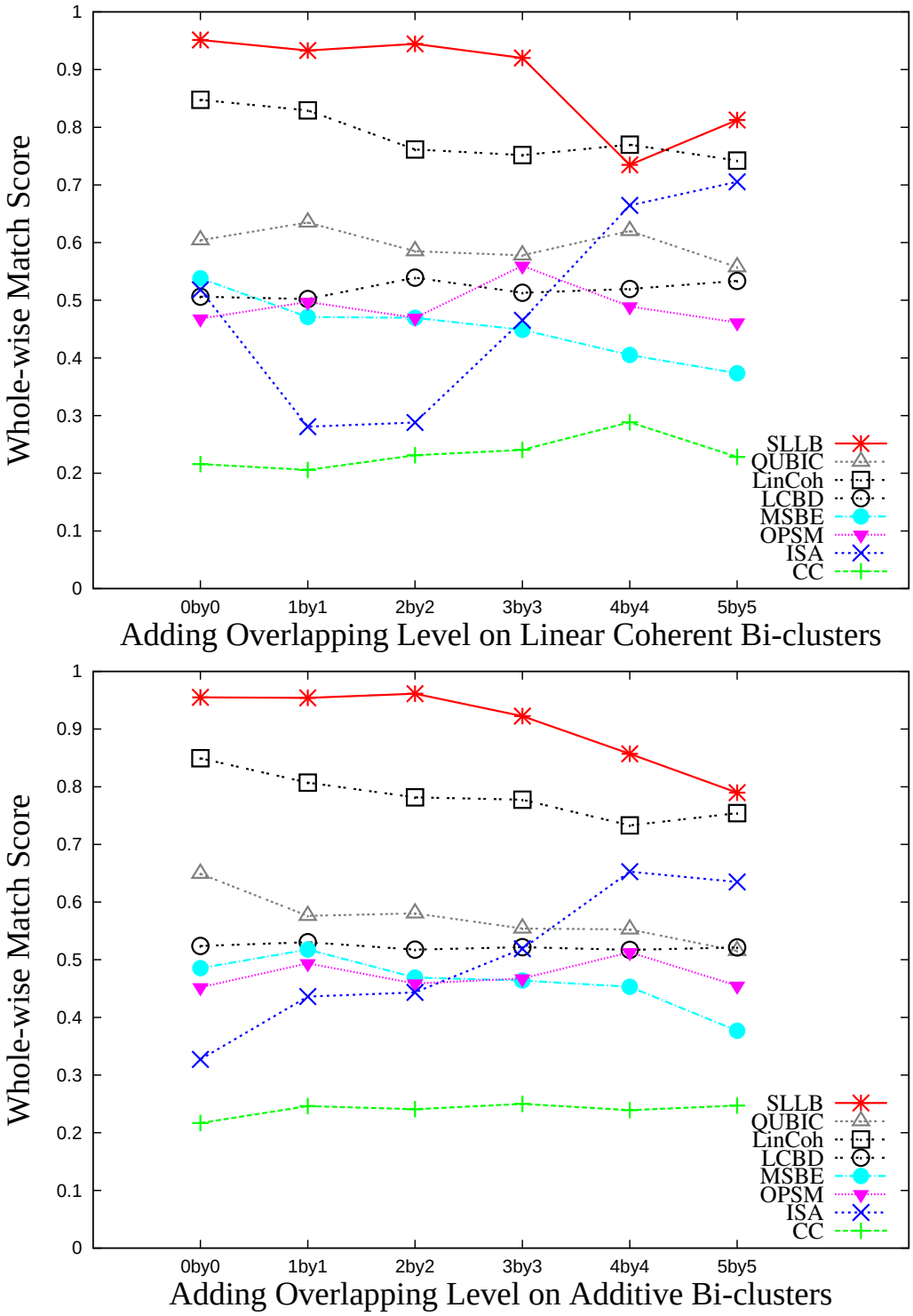


Figure C.3: The whole match scores of the eight algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the adding overlap model.

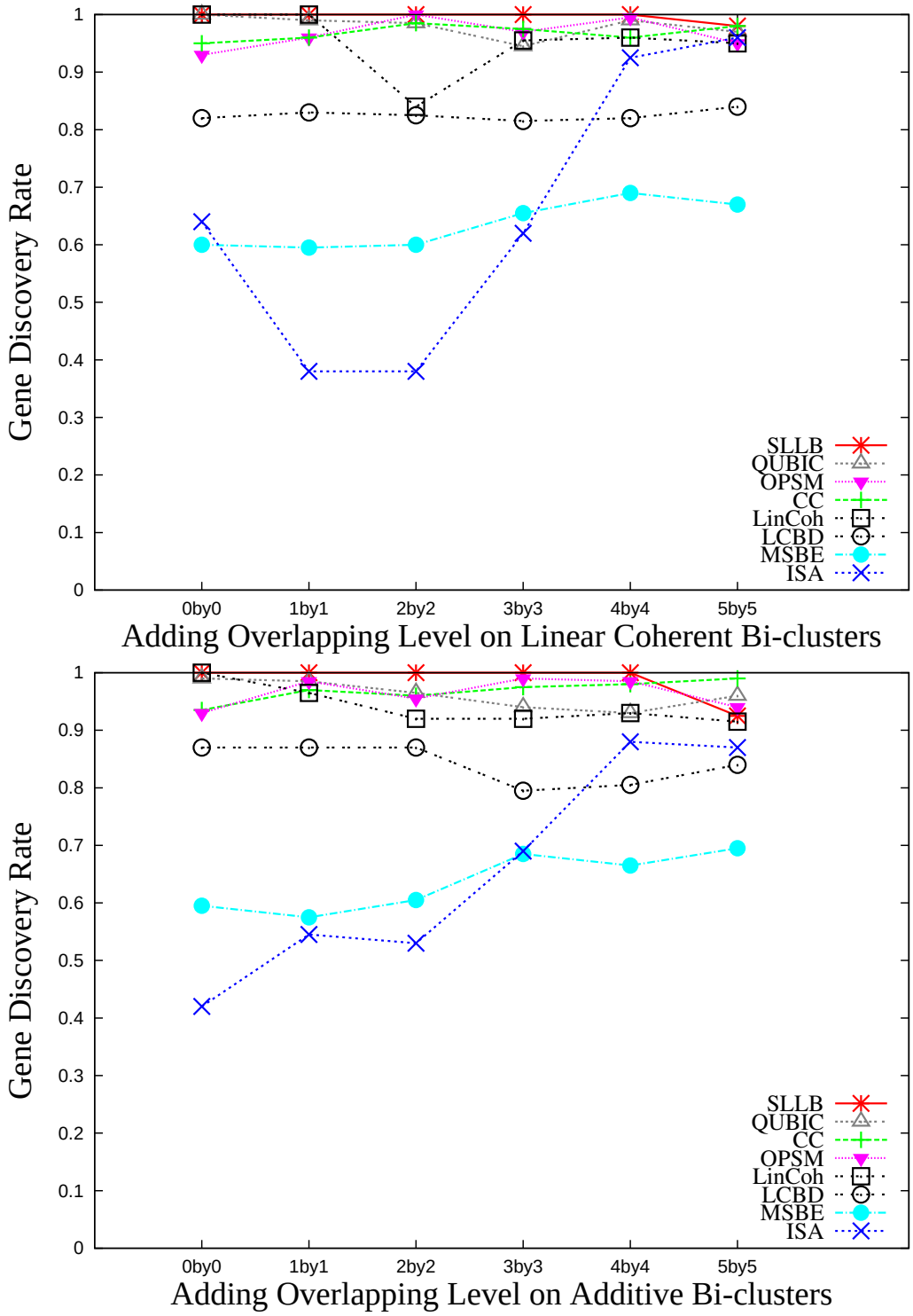


Figure C.4: The observation discovery rate of the eight algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the adding overlap model.

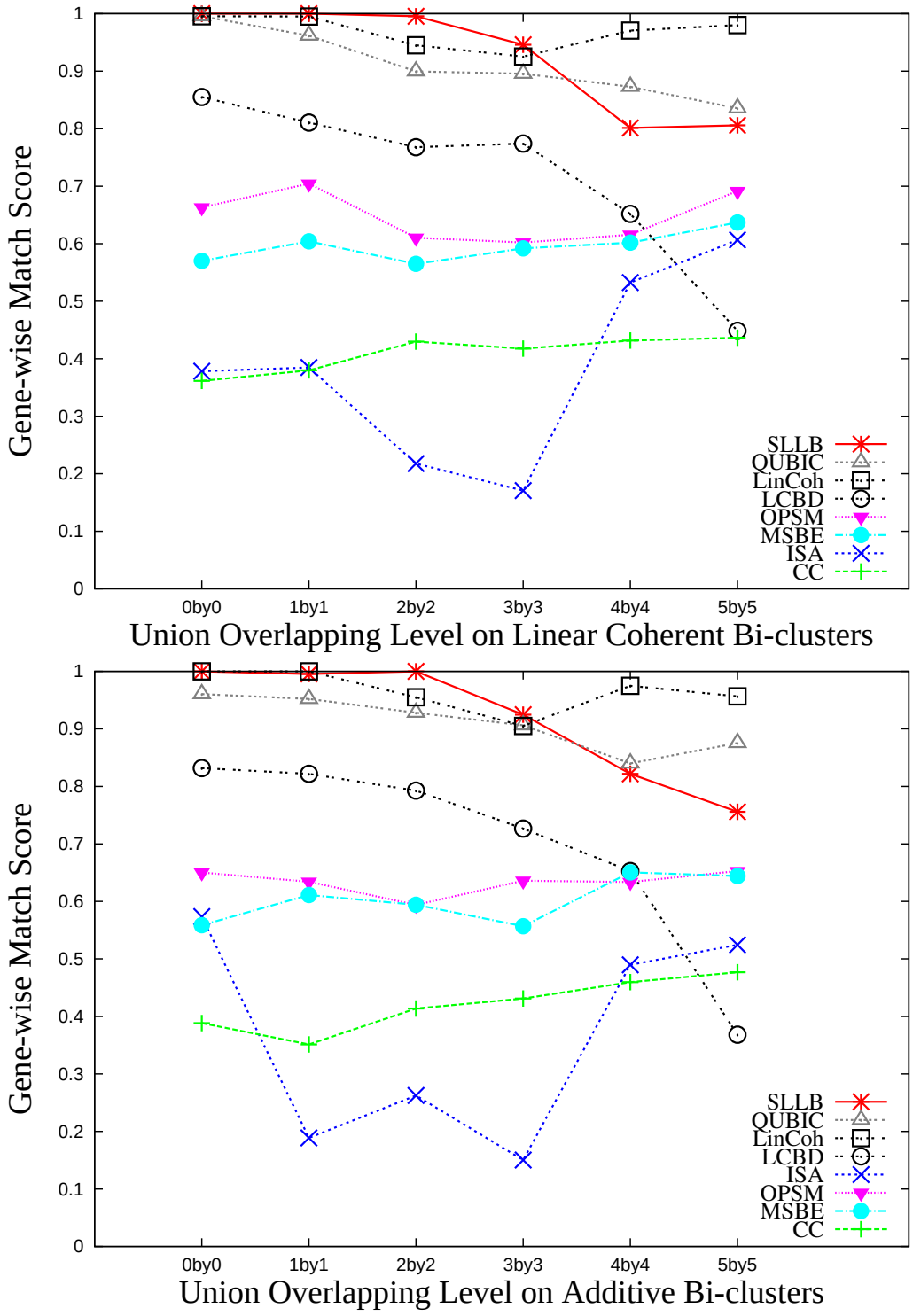


Figure C.5: The observation match scores of the eight algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the union overlap model.

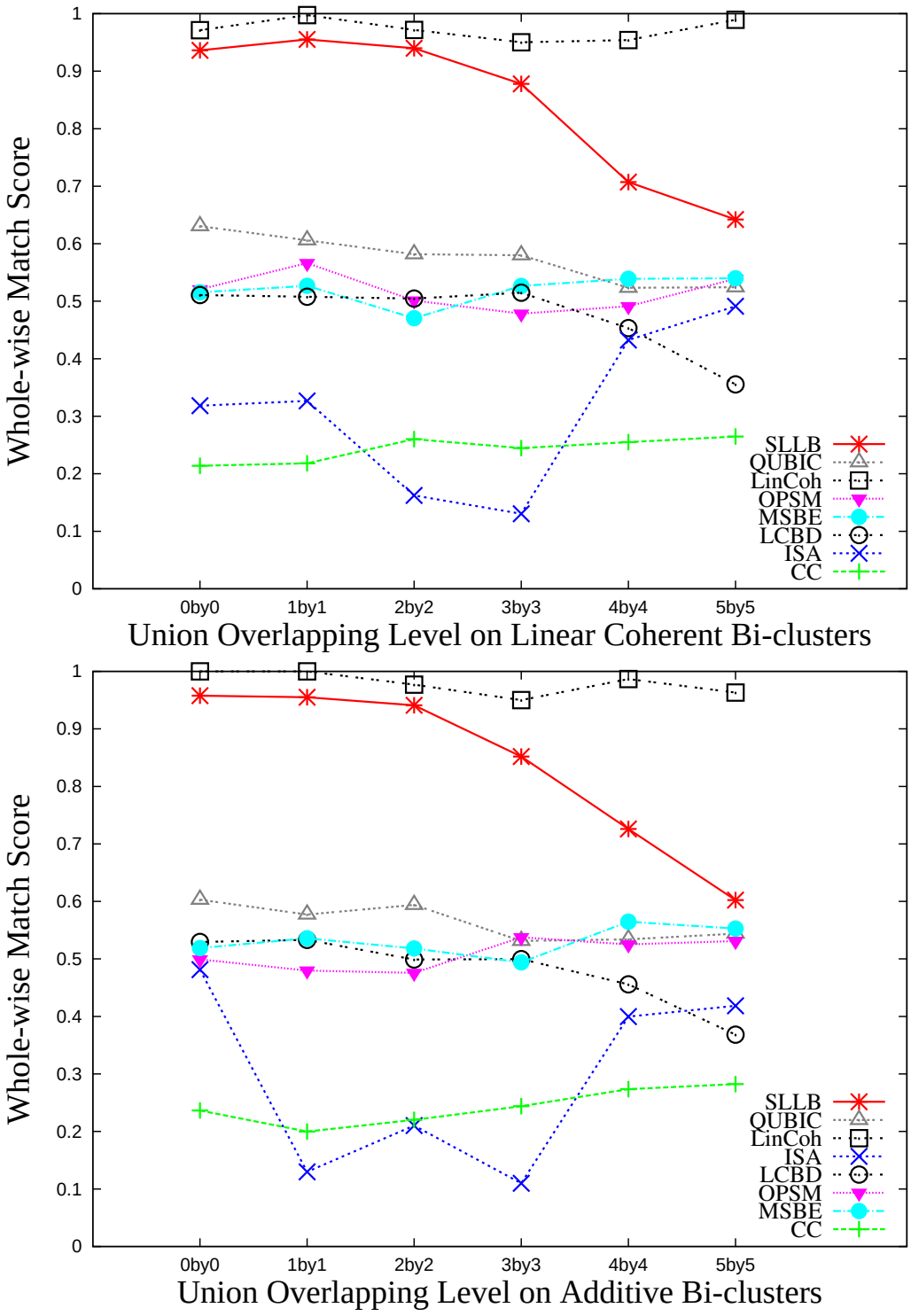


Figure C.6: The whole match scores of the eight algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the union overlap model.

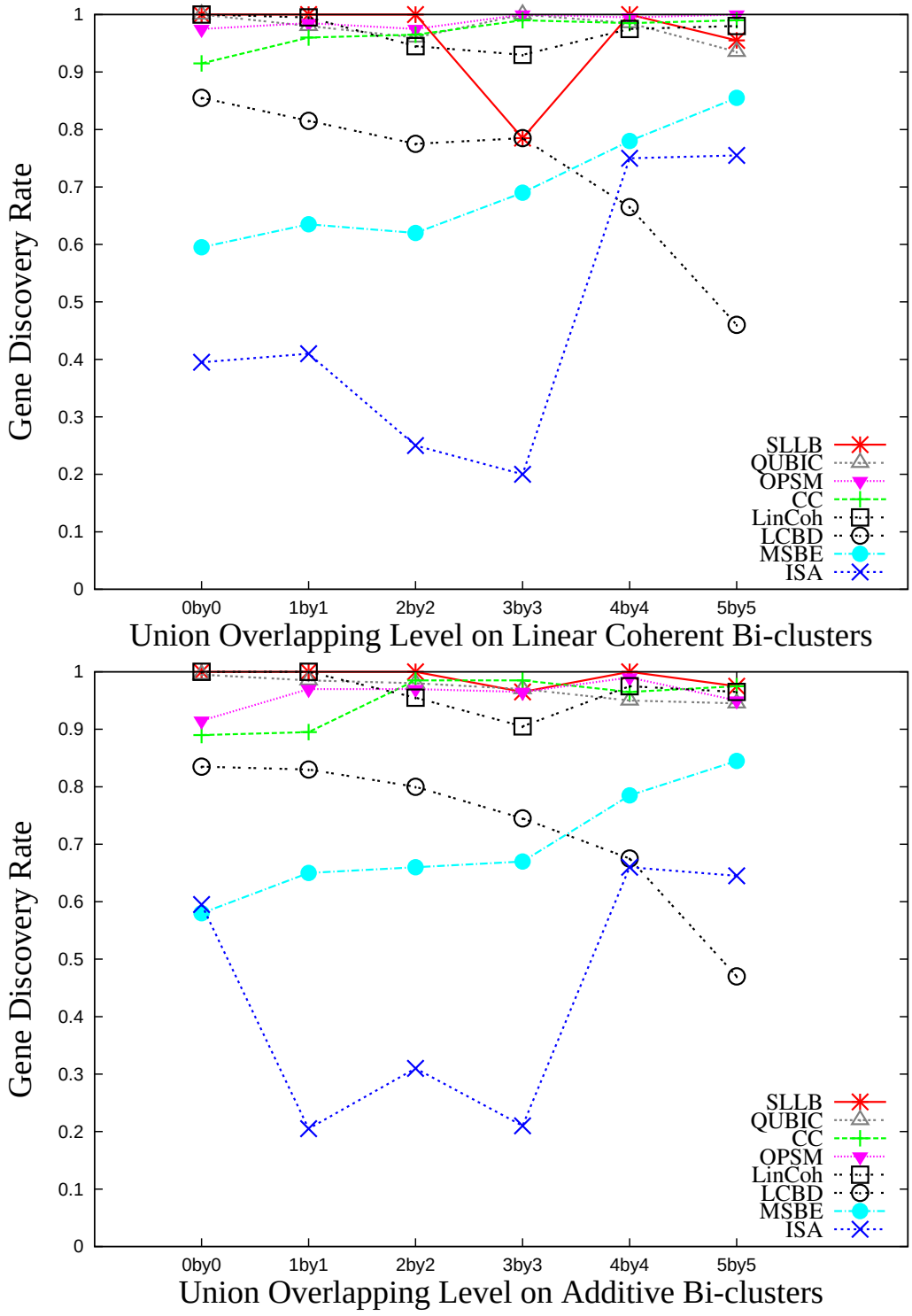


Figure C.7: The observation discovery rate of the eight algorithms for recovering the overlapping linear coherent and additive bi-clusters, under the union overlap model.

Table C.1: Statistics of different algorithms' bi-clustering results and the numbers of functional terms enriched on different databases.

	#Bi-clusters	$\mu_{\|gene\|}$	$\sigma_{\|gene\|}$	$\mu_{\|sample\|}$	$\sigma_{\|sample\|}$	#Terms enriched (GO, KEGG, MIPS/regulons)
Yeast:						
SLLB	101	83.40	36.40	85.56	25.98	3, 7, 5
QUBIC	100	119.08	59.69	23.21	6.21	3, 2, 3
LinCoh	100	61.84	38.43	133.09	18.09	5, 7, 5
LCBD	132	46.46	17.53	13.35	4.58	10, 6, 11
ISA	47	67	34.54	8.4	1.78	15, 13, 18
OPSM	14	423.29	728.95	9.07	5.14	1, 1, 2
MSBE	40	19.25	8.32	18.68	8.22	8, 4, 6
CC	10	297.7	304.18	60.8	23.46	6, 4, 8
E.Coli:						
SLLB	52	43.79	16.23	106.58	51.70	8, 11, 13
QUBIC	100	73.91	33.45	33.51	14.51	14, 8, 15
LinCoh	100	9.63	7.66	141.43	34.04	24, 24, 22
LCBD	155	485.05	366.63	15.37	22.95	23, 22, 33
ISA	34	124.21	42.18	13.88	6.11	11, 10, 13
OPSM	14	419.29	744.35	8.93	4.8	8, 4, 5
MSBE	9	82.67	18.1	80.22	19.18	1, 3, 4
CC	10	309.9	950.15	31.4	81.74	2, 2, 2

Appendix D

Feature selection SVM

(a). Semi-definite programming. By introducing two variables $\mathbf{a}$ and $\mathbf{b}$, I use the Lagrange multipliers to move the λ constraints to the objective function to get:

$$\min_{\gamma\geq 0, a\geq 0, b\geq 0} \beta_2 \mathbf{1}^T\gamma + \max_{\lambda} \lambda^T\mathbf{1} - \frac{1}{2}\lambda^T B\lambda + \mathbf{a}^T\lambda + \mathbf{b}^T(1-\lambda) \tag{D.1}$$

where $B = \frac{1}{\beta_1}\triangle(\mathbf{y})(\sum_j \gamma_j K_j)\triangle(\mathbf{y})$. Introducing δ to rewrite the $\min\max$ problem in $\min$ problem:

$$\begin{aligned} &\min_{\gamma\geq 0, \mathbf{a}\geq 0, \mathbf{b}\geq 0, \delta} \beta_2 \mathbf{1}^T\gamma + \delta \\ \textbf{s.t.}: \delta \geq &\max_{\lambda} \mathbf{b}^T\mathbf{1} - \frac{1}{2}\lambda^T B\lambda + \lambda^T(1+\mathbf{a}-\mathbf{b}) \end{aligned} \tag{D.2}$$

Let $\frac{\partial(\mathbf{b}^T\mathbf{1} - \frac{1}{2}\lambda^T B\lambda + \lambda^T(\mathbf{1}+\mathbf{a}+\mathbf{b}))}{\partial\lambda} = 0$, we get: $B\lambda = \mathbf{1}+\mathbf{a}-\mathbf{b}$. Substituting it to the δ constraint, we get:

$$\delta \geq \mathbf{b}^T\mathbf{1} + \frac{1}{2}(\mathbf{1}+\mathbf{a}-\mathbf{b})^T B^+(\mathbf{1}+\mathbf{a}-\mathbf{b})$$

Therefore, we can write the problem in the semi-definite programming form:

$$\begin{aligned} &\min_{\gamma\geq 0, \mathbf{a}\geq 0, \mathbf{b}\geq 0, \delta} \beta_2 \mathbf{1}^T\gamma + \delta \\ \textbf{s.t.}: &\begin{bmatrix} \frac{2}{\beta_1}\triangle(\mathbf{y})\sum_j(\gamma_j K_j)\triangle(\mathbf{y}) & (\mathbf{1}+\mathbf{a}-\mathbf{b}) \\ (\mathbf{1}+\mathbf{a}-\mathbf{b})^T & \delta - \mathbf{b}^T\mathbf{1} \end{bmatrix} \succcurlyeq 0. \end{aligned} \tag{D.3}$$

(b). Quadratic constraint linear programming (QCLP). According to Sion's $\min\max$ duality theorem (15) can be re-written as:

$$\max_{0\leq\lambda\leq\mathbf{1}} \min_{\gamma\geq 0} \lambda^T\mathbf{1} - \frac{1}{2\beta_1}\lambda^T\triangle(\mathbf{y})(\sum_j \gamma_j K_j)\triangle(\mathbf{y})\lambda + \beta_2\mathbf{1}^T\gamma, \tag{D.4}$$

By introducing a new variable υ to make γ unconstrained, we get:

$$\max_{0\leq\lambda\leq\mathbf{1}} \max_{\upsilon\geq 0} \min_{\gamma} \lambda^T\mathbf{1} - \frac{1}{2\beta_1}\lambda^T\triangle(\mathbf{y})(\sum_j \gamma_j K_j)\triangle(\mathbf{y})\lambda + \beta_2\mathbf{1}^T\gamma - \upsilon^T\gamma, \quad \text{(D.5)}$$

Let the term $\frac{1}{2\beta_1}\lambda^T\triangle(\mathbf{y})\sum_j(\gamma_j K_j)\triangle(\mathbf{y})\lambda = -\frac{1}{2\beta_1}S^T\gamma$, where $S_i = \lambda^T\triangle(\mathbf{y})K_i\triangle(\mathbf{y})\lambda$. Then let partial derivative of objective function with respect to γ equals to 0, we get $\beta_2\mathbf{1} - \frac{1}{2\beta_1}S - \upsilon = 0$, so that $\beta_2\mathbf{1} - \frac{1}{2\beta_2}S = \upsilon \geq 0 \Rightarrow S \leq 2\beta_1\beta_2\mathbf{1} \Rightarrow \lambda^T\triangle(\mathbf{y})K_i\triangle(\mathbf{y})\lambda \leq 2\beta_1\beta_2, \forall i$. So the original problem can be formulated in the following QCLP form which can be solved in QCLP solver:

$$\begin{aligned} &\max_{0\leq\lambda\leq 1} \mathbf{1}^T\lambda \\ \textbf{s.t.} : &\lambda^T\triangle(\mathbf{y})K_j\triangle(\mathbf{y})\lambda \leq 2\beta_1\beta_2, \forall i. \end{aligned} \quad \text{(D.6)}$$

(c). Generic constraint generation. Introduce a new variable δ into (15) to get:

$$\begin{aligned} &\min_{\gamma\geq 0,\delta} \delta \\ \textbf{s.t.} : \delta \geq \beta_2\mathbf{1}^T\gamma + \mathbf{1}^T\lambda - &\frac{1}{2\beta_1}\lambda^T\triangle(\mathbf{y})\sum_j(\gamma_j K_j)\triangle(\mathbf{y})\lambda \\ &0 \leq \lambda \leq 1 \end{aligned} \quad \text{(D.7)}$$

Then use the generic constraint generation algorithm to solve the above problem. Randomly generate a set of λ to get a set of constraints:

$$\begin{aligned} \delta &\geq F(\gamma, \lambda_1) \\ \delta &\geq F(\gamma, \lambda_2) \\ &\cdots \\ \delta &\geq F(\gamma, \lambda_k) \end{aligned} \quad \text{(D.8)}$$

Solve the $min_{\gamma\geq 0,\delta}\delta$ problem based on these constraint to get solution $\gamma^{(k)}, \lambda^{(k)}$. Then solve the problem $max_{0\leq\lambda\leq 1}F(\gamma^{(k)}, \lambda) - \delta^{(k)}$. If the solution is greater than a given threshold ϵ, add the new constraint based on the new λ, halt otherwise.