Hindawi Publishing Corporation
Journal of Biomedicine and Biotechnology
Volume 2012, Article ID 245968, 8 pages
doi:10.1155/2012/245968

Research Article

An Integrative Approach to Infer Regulation Programs in a Transcription Regulatory Module Network

Jianlong Qi,[1] Tom Michoel,[1] and Gregory Butler[2]

[1] *Freiburg Institute for Advanced Studies, University of Freiburg, Albertstraße 19, 79104 Freiburg im Breisgau, Germany*
[2] *Department of Computer Science and Software Engineering, Concordia University, 1455 de Maisonneuve Boulevard W, Montreal, QC, Canada H3G 1M8*

Correspondence should be addressed to Gregory Butler, gregb@encs.concordia.ca

Received 27 October 2011; Accepted 12 February 2012

Academic Editor: Yong Lim

The module network method, a special type of Bayesian network algorithms, has been proposed to infer transcription regulatory networks from gene expression data. In this method, a module represents a set of genes, which have similar expression profiles and are regulated by same transcription factors. The process of learning module networks consists of two steps: first clustering genes into modules and then inferring the regulation program (transcription factors) of each module. Many algorithms have been designed to infer the regulation program of a given gene module, and these algorithms show very different biases in detecting regulatory relationships. In this work, we explore the possibility of integrating results from different algorithms. The integration methods we select are union, intersection, and weighted rank aggregation. Experiments in a yeast dataset show that the union and weighted rank aggregation methods produce more accurate predictions than those given by individual algorithms, whereas the intersection method does not yield any improvement in the accuracy of predictions. In addition, somewhat surprisingly, the union method, which has a lower computational cost than rank aggregation, achieves comparable results as given by rank aggregation.

1. Introduction

There is a complex mechanism in cells that controls which genes are expressed. Transcription factors, which are a special type of protein and capable of regulating the expression of other genes by binding to their upstream regions, play a crucial role in this mechanism. Transcriptional regulatory relationships between transcription factors and their targets can be represented by a network, called a *transcription regulatory network*, where each vertex denotes a gene and each edge denotes a regulatory relationship.

Identifying transcription regulatory networks is critical, because it facilitates understanding biological processes in cells. Gene expression data, which measure mRNA levels of genes, are widely used for inferring transcription regulatory networks [1]. Bayesian networks, a type of probabilistic graphical model [2], have been proposed to infer transcription regulatory networks from gene expression data [3, 4]. Despite their success in learning regulatory networks, models inferred by the Bayesian networks tend to overfit the data because, in a gene expression dataset, the number of variables is normally large compared to the number of samples. To cope with this problem, Segal et al. [5] designed the module network method, which is a special type of Bayesian network algorithm. In this method, each module represents a set of variables that share (1) a single variable or a set of variables as their parents and (2) local distributions. Compared to standard Bayesian network algorithms, this design significantly reduces the number of parameters to be learned and consequently leads to more accurate inferences. The module network method has yielded promising results in learning regulatory networks [5–7].

Given a gene expression dataset and a list of candidate transcription factors, the process of learning module networks consists of two tasks: clustering genes into modules and inferring the regulation program (transcription factors) of each module. Segal et al. [5] designed an expectation-maximization-based [8] learning algorithm that alternates between these two tasks. Moreover, Joshi et al. [9] separated the two tasks, where they grouped genes into modules *before* learning the regulation program of each module. Experimental results showed that the separation improves the performance of the module network method in inferring regulatory networks.

Many techniques have been applied to infer the regulation program of a given gene module, that is, the second task in learning module networks, such as logistic regression [9], moderated *t*-statistics [10] from LIMMA [11], Gibbs sampling [12], and linear regression [13]. A common characteristic of these methods is that they are able to calculate the confidence (i.e., regulatory score) for the assignment of a transcription factor to a gene module, which is referred to as a regulator-module interaction. Consequently, their results can be sorted into an ordered list of regulator-module interactions according to their regulatory scores. The higher the ranking of a regulator-module interaction in the ordered list given by a method, the more confidence this method assigns to the interaction. In addition, since these methods resort to distinct techniques, they show very different biases in detecting regulatory relationships. For example, in the nitrogen utilization module in yeast, LeMoNe [9] favors regulatory relationships where transcription factors and genes are globally coexpressed, while the LIMMA-based method [10] favors regulatory relationships where transcription factors and genes are locally coexpressed. This suggests that integrating results from different regulation program learning algorithms can be a promising direction in better inferring regulatory networks [14].

In this work, we extend our previous work [10] by integrating its results with those given by two other learning algorithms [9, 13]. To the best of our knowledge, this is the first of such an attempt. The integration methods we select are union, intersection, and weighted rank aggregation [15]. Experimental results indicate that the union and weighted rank aggregation methods produce more accurate predictions than those given by individual algorithms, whereas the intersection method does not yield any improvement in the accuracy of predictions.

The rest of this paper is organized as follows: Section 2 describes the dataset, integration methods, and regulation program learning algorithms studied in this work. Section 3 presents experimental results. Section 4 summarizes the main results and discusses future work.

2. Material and Method

2.1. Data Set and Reference Database. The yeast stress dataset has been used as a benchmark to validate the performance of module network learning algorithms [5, 9]. This dataset consists of 173 arrays and measures the budding yeast's response to a panel of diverse environmental stresses [16]. The conditions covered by the dataset consist of temperature shocks, amino acid starvation, nitrogen source depletion, and so on. In previous work [5, 17], 2355 differentially expressed genes in the dataset were selected and these genes were clustered into 69 gene modules. Genes in the modules show functional enrichment. For example, a module for nitrogen utilization was obtained. This module consists of 47 genes mostly involved in two pathways: the methionine pathway (regulated by MET28 and MET32), and the nitrogen catabolite repression (NCR) system (regulated by GLN3, GZF3, DAL80, and GAT1). Both pathways relate to the process by which the budding yeast uses the best available nitrogen source in the environment [18, 19].

In this work, we apply three algorithms [9, 10, 13] to infer regulators of these modules using a list of 321 transcription factors prepared by Segal et al. [5] as candidate transcription factors. Then, we integrate the results of these algorithms by methods described in Section 2.2. The regulatory relationships recorded in YEASTRACT [20] (released on April 27, 2009) are used as the reference database to validate results given by individual algorithms and our integration methods.

2.2. Integration Methods. We apply union, intersection, and weighted rank aggregation integration methods to integrate results from different regulation program learning algorithms. The union and intersection methods are straightforward. The former determines the ranking of a regulator-module interaction using the highest ranking given by all candidate learning algorithms. In contrast, the latter determines the ranking of an interaction using the lowest ranking. For example, given a regulator-module interaction, which is the 1st, 3rd, and 5th items in rankings given by three individual learning algorithms, respectively, the union method assigns the 1st as its ranking, while the intersection method assigns the 5th as its ranking. After determining the ranking of each interaction, these methods can each produce an ordered list of regulator-module interactions by sorting interactions by their rankings.

In comparison, the weighted rank aggregation method [15] is much more computationally intensive than the union and intersection methods. Given a set of learning algorithms M, this integration algorithm searches for an ordered list δ^*, that is, simultaneously as close as possible to the list produced by each algorithm in M. Let $L_m = (A_1^m, A_2^m, \dots, A_k^m)$ represent an ordered list of k regulator-module interactions produced by the algorithm m. Let $r^m(A)$ denote the rank of the interaction A under m. Finally, let $m(i)$ $(i = 1, 2, \dots, k)$ denote the P value (weight) that algorithm m assigns to the interaction ranked at the ith position in the ordered list. This can be represented by the following minimization problem:

$$\delta^* = \arg\min \Phi(\delta), \tag{1}$$

where

$$\Phi(\delta) = \sum_{m \in M} d(\delta, L_m) \tag{2}$$

represents the sum of the distances between an ordered list δ and the lists from all algorithms. The distance between δ and L_m is determined by the weighted Spearman's footrule distance:

$$d(\delta, L_m) = \sum_{A \in L_m \cup \delta} \left| m\left(r^{\delta}(A)\right) - m(r^m(A)) \right| \times \left| r^{\delta}(A) - r^m(A) \right|. \tag{3}$$

To determine δ^*, we apply the cross-entropy Monte Carlo algorithm [21].

2.3. Regulation Program Learning Algorithms. We select LeMoNe [9], Inferelator [13], and the LIMMA-based method [10], as candidate regulation program learning algorithms. In this section, we describe how to apply these algorithms to the yeast stress dataset. In addition, in order to apply the weighted rank aggregation to integrate their results, for each algorithm, we also define how to calculate the P value for the assignment of a regulator to a module.

2.3.1. LeMoNe. For each gene module in the yeast stress dataset, LeMoNe [9] sampled 10 regression trees and then calculated regulatory scores for assigning transcription factors to this module based on these trees. Regulatory scores of all regulator-module interactions were downloaded from the supplementary website of [9].

We calculate the LeMoNe-based P value for the assignment of a regulator r to a module as follows. First, given a regression tree T of this module, we define the P value of the split with r and a splitting value z at an internal node t in T (i.e., $P\ \text{value}_{(t)}(r,z)$) as the probability of observing a split with a higher average prediction probability than this split at the node t. The average prediction probability of a split is defined as in equation (4) of [9]. Then, the P value for assigning r to t is defined as:

$$P\ \text{value}_{(t)}(r) = \frac{w_t}{|Z|}\sum_{z\in Z} P\ \text{value}_{(t)}(r,z), \tag{4}$$

where w_t is the number of experimental conditions in t divided by the total number of conditions in the data, and Z represents the set of possible splitting values for r in t. Furthermore, given a set of regression trees F, the LeMoNe-based P value for assigning r to this module can be calculated as:

$$P\ \text{value}(r) = \frac{1}{|F|}\sum_{T\in F}\sum_{t\in T} P\ \text{value}_{(t)}(r). \tag{5}$$

2.3.2. Inferelator. Inferelator [13] uses linear regression and variable selection to identify transcription factors of gene modules. In each gene module in the yeast stress dataset, we fit a linear model to the mean of the module's genes in each condition using the 321 candidate transcription factors as predictor variables. The regulatory score for assigning a regulator to the module is decided by the absolute value of the regulator's regression coefficient in the fitted model.

The Inferelator-based P value for the assignment of a regulator to a module is defined as follows. First, we permute the values of the expression value matrix from the row direction (gene). Second, we apply Inferelator to the permuted dataset using the original gene modules. Third, we fit the distribution of nonzero coefficients obtained from the permuted dataset by the Weibull distribution defined as:

$$\text{pdf}(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, \tag{6}$$

with $k = 0.889$ and $\lambda = 0.015$ (Figure 1). Last, we define the Inferelator-based P value for a regulator-module interaction with a regulatory score S as the probability of observing a value more than S from the Weibull distribution (6).

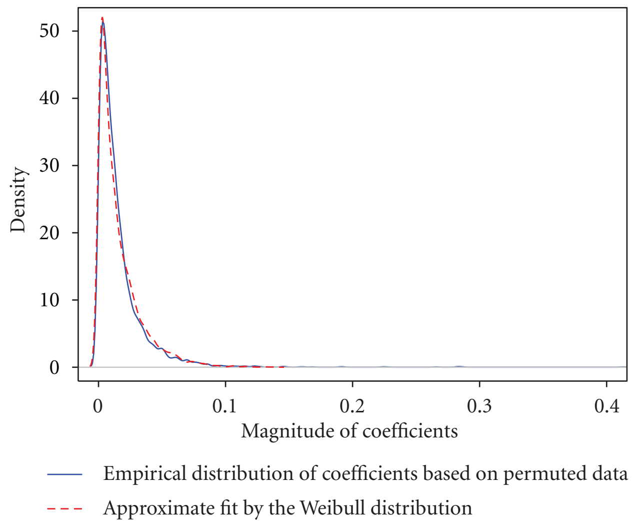

Figure 1: Probability density function of coefficients (regulatory scores) based on permuted data and the approximated fit by the Weibull distribution. The solid line denotes the empirical probability density function of regression coefficients obtained by Inferelator in the permuted expression data, while the dotted line denotes the probability density function of the Weibull distribution (6) with $k = 0.889$ and $\lambda = 0.015$.

2.3.3. LIMMA-Based Method. In our previous work [10], moderated t-statistics proposed in LIMMA [11] were applied to infer transcription factors of gene modules. For each gene module in the yeast dataset, ten condition clusterings were sampled by a two-way clustering algorithm [17]. Then the regulatory score for assigning a transcription factor to this module was calculated by summing the transcription factor's standardized moderated t-statistics based on the sampled condition clusterings.

We next describe how to define the method's P value for the assignment of a transcription factor to a module. First, we randomly generate ten condition clusterings, each of which consisted of two clusters. We then calculate the regulatory score for each candidate transcription factor based on these randomly generated clusterings. Moreover, we record the regulatory score of a randomly selected transcription factor. The above process is repeated to obtain 100,000 randomly generated regulatory scores. Last, the probability density function of these randomly generated scores is approximated by the stretched exponentials [22] defined as:

$$\text{pdf}(x) = \begin{cases} h_{\max}\exp\left[-b_r(x - x_{\max})^{c_r}\right], & \text{for } x \ge x_{\max}, \\ h_{\max}\exp\left[-b_l(x_{\max} - x)^{c_l}\right], & \text{for } x < x_{\max}, \end{cases} \tag{7}$$

with $h_{\max} = 0.127$, $b_r = 0.024$, $b_l = 0.083$, $c_r = 2.45$, $c_l = 1.70$, and $x_{\max} = -0.050$. As shown in Figure 2, the approximated fit is very close to the empirical distribution of the randomly generated regulatory scores, so the P value for a regulator-module interaction with a regulatory score S can be defined as the probability of observing a value more than S from the approximated fit.

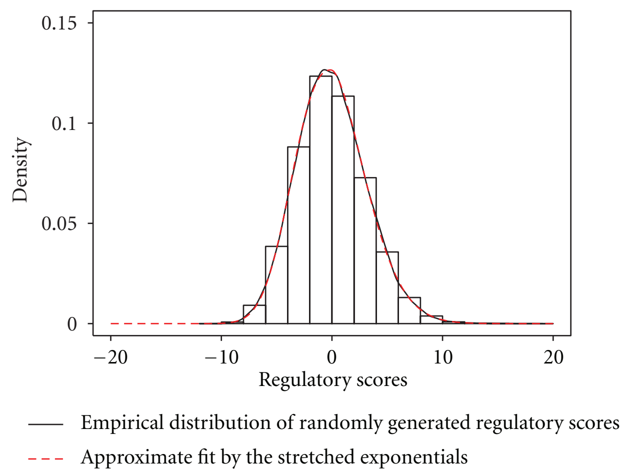

Figure 2: Probability density function of randomly generated regulatory scores and the approximated fit by the stretched exponentials. The solid line denotes the empirical probability density function of regulatory scores obtained by the LIMMA-based method based on randomly generated condition clusterings, while the dotted line denotes the stretched exponentials (7) with $h_{\max} = 0.127$, $b_r = 0.024$, $b_l = 0.083$, $c_r = 2.45$, $c_l = 1.70$, and $x_{\max} = -0.050$.

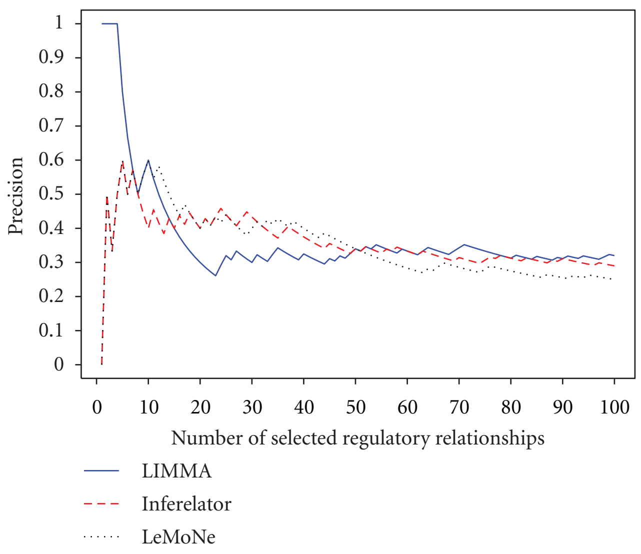

Figure 3: Comparison of precision of three candidate learning algorithms. For each algorithm, the figure shows the precision (8) when the top i ($i = 1, 2, \ldots, 100$) regulator-module interactions in the rank given by the algorithm are selected.

Note that the assignment of a regulator to a module is associated with a P value for each regulation program learning algorithm, and the P value is required by the weighted rank aggregation method to integrate results from different learning algorithms. In contrast, the assignment is also associated with a P value based on records in the reference database, YEASTRACT. This P value is calculated by the hypergeometric distribution and is used to evaluate the performance of individual learning algorithms and integration methods.

3. Results and Discussion

3.1. Results of Individual Learning Algorithms. We applied each regulation program learning algorithm described in Section 2.3 to calculate the regulatory score for assigning a regulator to a module. Then we sorted all of its regulatory scores between 321 candidate transcription factors and 69 modules in descending order. This led to an ordered list of 22,149 regulator-module interactions for each method.

In addition, for each regulator-module interaction, we used the hypergeometric distribution to calculate the P value of this interaction, using regulatory relationships in YEASTRACT as the reference database. This P value is based on the number of genes regulated by the regulator in the dataset, the number of genes regulated by the regulator in the module, and the number of genes in the module, and is used to determine if the regulator-module interactions is a true positive.

Moreover, for a given ordered list of regulator-module interactions, we define the precision of the top i items in this ordered list as:

$$P(i) = \frac{T(i)}{i}, \tag{8}$$

where $T(i)$ denotes the number of interactions with P values less than 0.05 in the top i items (i.e., the number of true positives among these i interactions).

In Figure 3, we show the precision of the top i regulator-module interactions ($i = 1, 2, \ldots, 100$) in the ordered lists obtained by Inferelator, LeMoNe, and the LIMMA-based method. When less than 20 interactions are selected, the LIMMA-based method outperforms the other two methods. Most true positives given by LIMMA-based method are for the module of nitrogen utilization. However, Inferelator and LeMoNe outperform the LIMMA-based method when the number of selected interactions is in the range of 20 and 50. In addition, when more than 50 interactions are selected, the three methods show similar performance in the yeast dataset.

3.2. Results for the Weighted Rank Aggregation. The weighted rank aggregation method searches for a synthesized list that is simultaneously as close as possible to the ordered lists from LeMoNe, Inferelator, and the LIMMA-based method. However, it is not feasible to directly apply this integration method on a list with 22,149 interactions due to the extensive computational workload. Hence, we resort to a tradeoff by integrating the top k ($k \leq 22,149$) interactions in the ordered lists given by these algorithms. This is, for a given ordered list and k, interactions ranked lower than k (i.e., $k+1, k+2, \ldots, 22,149$), are associated with a same weight (P value) of one. The larger k, the closer the list produced by the rank aggregation is to the lists given by the three candidate algorithms, but the rank aggregation costs more computation time. For example, it takes 12 hours and 48 hours for $k = 75$ and $k = 100$, respectively, on a HP Rackmount server with AMD Opteron processors (×86, 64 bit, dual core) and 16 GB memory.

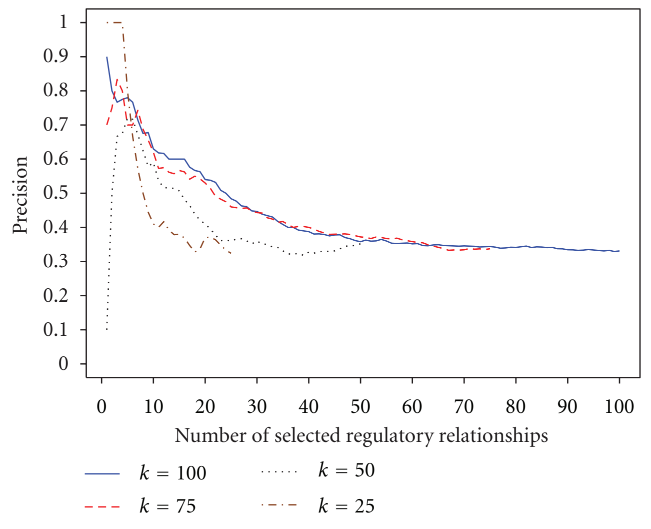

FIGURE 4: Comparison of precision given by the rank aggregation at $k = 25, 50, 75$, and 100.

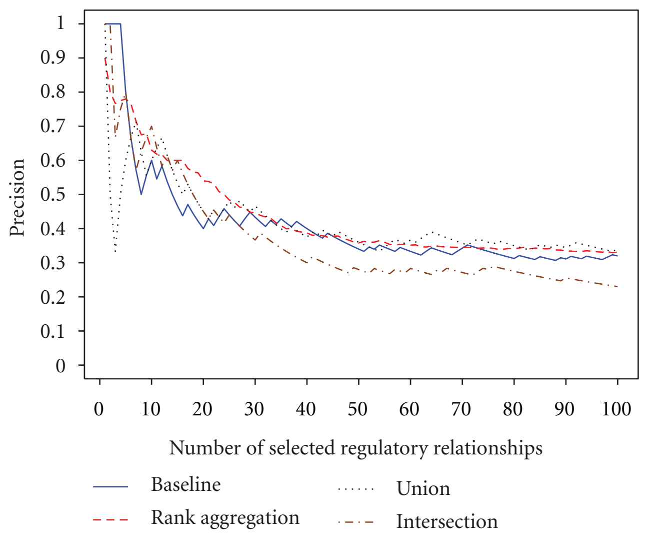

FIGURE 5: Comparison between precisions of integration methods and baseline precisions. For each of the three integration methods, the figure shows the precision (8) when the top i ($i = 1, 2, \ldots, 100$) regulator-module interactions in the rank given by the integration method are selected. For a given i, the baseline precision denotes the maximum of precisions obtained by individual learning algorithms (9).

In order to select a proper value for k in the yeast dataset, we applied the rank aggregation ten times for $k = 25, 50, 75, 100$, respectively. For each k, this led to 10 ordered lists, and we calculated the average of the precision of the top i ($i = 1, 2, \ldots, k$) interactions in these ten lists. As shown in Figure 4, when k increases from 25 to 50 and then to 75, the precisions obtained by the rank aggregation method are improved, but the precisions at $k = 75$ and $k = 100$ are about the same. This indicates that after k reaches 75, considering more interactions from the ordered lists of the candidate algorithms can no longer improve the performance of the rank aggregation method. Hence, k is set to 100 in our tests using the yeast dataset.

3.3. Comparison of Integration Methods and Individual Algorithms. In this section, we compare the performance of integration methods with individual algorithms. In order to make the comparison clear, for a given i ($i = 1, 2, \ldots, 100$), we define the *baseline* precision as that obtained by selecting the maximum of the precisions of the top i interactions given by all individual algorithms. That is, given a set of individual learning algorithms M, it is determined as:

$$P^*(i) = \max_{m \in M}\left(p_m(i)\right), \tag{9}$$

where $p_m(i)$ denotes the precision of top i interactions in the ordered list given by algorithm m (8). Note that baseline precision represents an upper optimistic bound that cannot be achieved by individual algorithms as we can only use one of them at a time. Hence, even if the precision obtained by an integration method is only comparable to baseline precision, it still shows that this integration method yields a better overall performance than those of individual algorithms.

We compare the precision of the top 100 predictions from the three integration methods with baseline precisions (Figure 5). The union and rank aggregation methods generate better or similar results compared to the baseline precision. In addition, somewhat surprisingly, the union method, which has a lower computational cost than rank aggregation, achieves comparable results as given by rank aggregation. The first twenty interactions from union and rank aggregation are shown in Tables 1 and 2, respectively.

On the other hand, we observe that baseline precisions are generally better than precisions given by the intersection method. The intersection method sorts interactions by their lowest rankings from all candidate algorithms, so it tends to assign interactions with moderate confidences from all algorithms with high ranks. We speculate that this may have affected its performance. For example, as shown in Table 3, its first 20 interactions include several not highly ranked by any algorithm, such as the twelfth interaction (RDS2 to module 16) ranked 219th, 125th, and 273rd by the LIMMA-based method, LeMoNe, and Inferelator, respectively; and the eighteenth interaction (DAL81 to module 58) ranked as 199th, 310th, and 190th by the LIMMA-based method, LeMoNe, and Inferelator, respectively.

We also compare areas under precision curves for the top 100 predictions given by the integration methods and individual learning algorithms (Table 4). The union and weighted rank aggregation methods achieve better results than those from the individual learning algorithms, but the intersection method only yields a comparable result with the individual learning algorithms. These results indicate that we should be cautious to apply the intersection method to integrate results from algorithms of different natures.

Table 1: Top twenty regulator-module interactions as given by the union method. *Records represent true positives at the P value threshold of 0.05.

Rank	Regulator-module	P value	Ranks from individual algorithms		
			LIMMA	LeMoNe	Inferelator
1	DAL80-11*	$3.47e-10$	1	130	88
	IME4-46	$1.00e+00$	48	1	2906
	HAP1-13	$1.00e+00$	3076	169	1
4	HAP4-7*	$1.67e-30$	50	9	2
	MET32-11*	$1.21e-13$	2	30	7
	DAL80-51*	$0.00e+00$	4	2	10281
7	PHD1-36*	$5.20e-03$	3	342	5
	HAP4-30	$5.33e-02$	23	3	32
	MET32-27	$1.00e+00$	9680	3702	3
10	TOS8-24*	$1.45e-02$	49	4	111
	GAT1-59*	$2.55e-03$	802	5059	4
12	XBP1-10*	$1.08e-02$	59	5	602
	DAL82-48	$1.00e+00$	5	49	7771
14	UGA3-11	$2.35e-01$	6	509	59
	USV1-28	$1.00e+00$	190	6	10281
	SKO1-57	$1.00e+00$	4663	10026	6
17	GAT1-11*	$4.24e-05$	34	7	28
	ACA1-48	$1.00e+00$	7	1048	7773
19	PDR3-13	$3.90e-01$	12	8	10281
	DAL80-40	$1.00e+00$	8	367	27

Table 2: Top twenty regulator-module interactions as given by the weighted rank aggregation method. *Records represent true positives at the P value threshold of 0.05.

Rank	Regulator-module	P value	Ranks from individual algorithms		
			LIMMA	LeMoNe	Inferelator
1	HAP4-7*	$1.67e-30$	50	9	2
2	GAT1-11*	$4.24e-05$	34	7	28
3	MET28-11*	$5.92e-10$	27	24	16
4	HAP4-30	$5.33e-02$	23	3	32
5	MET32-11*	$1.21e-13$	2	30	7
6	DAL80-51*	$0.00e+00$	4	2	10281
7	DAL81-6	$6.67e-01$	21	14	1193
8	PDR3-13	$3.90e-01$	12	8	10281
9	PHD1-36*	$5.20e-03$	3	342	5
10	IME4-46	$1.00e+00$	48	1	2906
11	TOS8-24*	$1.45e-02$	49	4	111
12	GAL80-41*	$7.45e-09$	7107	10	52
13	DAL81-55	$4.17e-01$	14	16	285
14	MET32-40*	$1.00e-02$	52	5329	37
15	CUP2-10*	$1.12e-02$	553	32	18
16	YAP5-51	$1.00e+00$	37	746	40
17	XBP1-10*	$1.08e-02$	59	5	602
18	YAP6-25	$8.63e-02$	26	26	5444
19	UGA3-40	$1.00e+00$	22	33	93
20	UGA3-11	$2.35e-01$	6	509	59

Table 3: Top twenty regulator-module interactions as given by the intersection method. *Records represent true positives at the P value threshold of 0.05.

Rank	Regulator-module	P value	Ranks from individual algorithms		
			LIMMA	LeMoNe	Inferelator
1	MET28-11*	$5.92e-10$	27	24	16
2	MET32-11*	$1.21e-13$	2	30	7
3	HAP4-30	$5.33e-02$	23	3	32
4	GAT1-11*	$4.24e-05$	34	7	28
5	HAP4-7*	$1.67e-30$	50	9	2
6	OAF1-22	$5.37e-02$	73	45	87
7	UGA3-40	$1.00e+00$	22	33	93
8	TOS8-24*	$1.45e-02$	49	4	111
9	DAL80-11*	$3.47e-10$	1	130	88
10	GAL4-22*	$7.44e-03$	155	85	115
11	MET32-51	$1.00e+00$	28	224	113
12	RDS2-16	$1.00e+00$	219	125	273
13	YRR1-24*	$4.06e-04$	273	21	230
14	DAL81-55	$4.17e-01$	14	16	285
15	GZF3-11*	$8.34e-04$	31	188	297
16	TOS8-49	$5.55e-02$	87	308	85
17	SWI4-55	$3.67e-01$	39	63	310
18	DAL81-58	$1.00e+00$	199	310	190
19	BAS1-63	$1.00e+00$	33	88	312
20	IME4-3	$4.52e-01$	317	58	245

Table 4: Comparison of areas under precision curves for the top 100 predictions given by the integration methods and individual learning algorithms. The precision curves for the integration methods are shown in Figure 5, while the precision curves for the individual learning algorithms are shown in Figure 3.

	Integration methods			Individual algorithms		
	Rank aggregation	Union	Intersection	LIMMA	Inferelator	LeMoNe
Area under curve	42.63	41.12	36.09	36.72	35.79	35.18

4. Conclusions

A metalearner approach was applied to infer transcription factors of coexpressed gene modules in a yeast stress dataset, with the regulatory relationships recorded in YEASTRACT as the gold standard. We integrated the predictions of three existing inference techniques [9, 10, 13] by three different methods: union, intersection, and weighted rank aggregation. Experimental results show that integrated predictions based on union or rank aggregation have higher precision than any of the individual methods. The justification of this work is that the results generated by different algorithms are not identical and often have clearly different influences from the datasets used. The experiments confirm our expectation that integrating the output of several algorithms results in higher quality predictions. To the best of our knowledge, this is the first such an attempt. Consequently, this work may point out a promising direction for module network learning.

An interesting extension of this work is to investigate if integrating results from more algorithms can lead to even better performance. In particular, we expect that when more algorithms are combined, we may see significant difference between the union and weighted rank aggregation methods. The experiments in this work are conducted on a yeast dataset, and results are validated by the regulatory relationships recorded in YEASTRACT, which does not represent a complete reference database of the regulatory network in the yeast. Hence, another direction for future work is to perform experiments on expression data from other species (e.g., *E. coli* [23]) to verify if results are consistent with those we obtained in the yeast dataset. In addition, we are interested in performing experiments on synthetic datasets (e.g., DREAM [24]), where complete reference networks are available.

References

[1] J. Tegnér, M. K.S. Yeung, J. Hasty, and J. J. Collins, "Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 10, pp. 5944–5949, 2003.

[2] D. Heckerman, "A tutorial on learning with Bayesian networks," in *Learning in Graphical Models*, pp. 301–354, MIT Press, 1999.

[3] N. Friedman, M. Linial, and I. Nachman, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[4] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich probabilistic models for gene expression," *Bioinformatics*, vol. 17, no. 1, supplement 1, pp. S243–S252, 2001.

[5] E. Segal, M. Shapira, A. Regev et al., "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data," *Nature Genetics*, vol. 34, no. 2, pp. 166–176, 2003.

[6] J. Li, Z. J. Liu, Y. C. Pan et al., "Regulatory module network of basic/helix-loop-helix transcription factors in mouse brain," *Genome Biology*, vol. 8, no. 11, article no. R244, 2007.

[7] E. Bonnet, T. Michoel, and Y. van de Peer, "Prediction of a gene regulatory network linked to prostate cancer from gene expression, microRNA and clinical data," *Bioinformatics*, vol. 26, no. 18, Article ID btq395, pp. i638–i644, 2010.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[9] A. Joshi, R. De Smet, K. Marchal, Y. Van de Peer, and T. Michoel, "Module networks revisited: computational assessment and prioritization of model predictions," *Bioinformatics*, vol. 25, no. 4, pp. 490–496, 2009.

[10] J. Qi, T. Michoel, and G. Butler, "Applying linear models tolearn regulation programs in a transcription regulatory module network," in *Proceedings of the 9th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pp. 37–47, 2011.

[11] G. K. Smyth, "Linear models and empirical Bayes methods for assessing differential expression in microarray experiments," *Statistical Applications in Genetics and Molecular Biology*, vol. 3, article3, 2004.

[12] J. Qi, T. Michoel, and G. Butler, "A regression tree-based Gibbs sampler to learn the regulation programs in a transcription regulatory module network," in *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 206–215, 2010.

[13] R. Bonneau, D. J. Reiss, P. Shannon et al., "The inferelator: an algorithn for learning parsimonious regulatory networks from systems-biology data sets de novo," *Genome Biology*, vol. 7, no. 5, article no. R36, 2006.

[14] T. Michoel, R. De Smet, A. Joshi, Y. Van de Peer, and K. Marchal, "Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks," *BMC Systems Biology*, vol. 3, no. 1, article no. 49, 2009.

[15] V. Pihur, S. Datta, and S. Datta, "Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach," *Bioinformatics*, vol. 23, no. 13, pp. 1607–1615, 2007.

[16] A. P. Gasch, P. T. Spellman, C. M. Kao et al., "Genomic expression programs in the response of yeast cells to environmental changes," *Molecular Biology of the Cell*, vol. 11, no. 12, pp. 4241–4257, 2000.

[17] A. Joshi, Y. Van de peer, and T. Michoel, "Analysis of a Gibbs sampler method for model-based clustering of gene expression data," *Bioinformatics*, vol. 24, no. 2, pp. 176–183, 2008.

[18] B. Magasanik and C. A. Kaiser, "Nitrogen regulation in *Saccharomyces cerevisiae*," *Gene*, vol. 290, no. 1-2, pp. 1–18, 2002.

[19] T. S. Cunningham, R. Rai, and T. G. Cooper, "The level of DAL80 expression down-regulates GATA factor-mediated transcription in *Saccharomyces cerevisiae*," *Journal of Bacteriology*, vol. 182, no. 23, pp. 6584–6591, 2000.

[20] P. T. Monteiro, N. D. Mendes, M. C. Teixeira et al., "YEASTRACT-DISCOVERER: new tools to improve the analysis of transcriptional regulatory associations in *Saccharomyces cerevisiae*," *Nucleic Acids Research*, vol. 36, supplement 1, pp. D132–D136, 2008.

[21] V. Pihur, S. Datta, and S. Datta, "RankAggreg, an R package for weighted rank aggregation," *BMC Bioinformatics*, vol. 10, no. 1, article no. 62, 2009.

[22] G. Stolovitzky, R. J. Prill, and A. Califano, "Lessons from the DREAM2 challenges: a community effort to assess biological network inference," *Annals of the New York Academy of Sciences*, vol. 1158, pp. 159–195, 2009.

[23] J. J. Faith, B. Hayete, J. T. Thaden et al., "Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles," *PLoS Biology*, vol. 5, no. 1, pp. 54–66, 2007.

[24] R. J. Prill, D. Marbach, J. Saez-Rodriguez et al., "Towards a rigorous assessment of systems biology models: the DREAM3 challenges," *PLoS One*, vol. 5, no. 2, Article ID e9202, 2010.