Identifying Gene Regulatory Networks from Experimental Data

Ting Chen*, Vladimir Filkov†, Steven S. Skiena‡

1 Introduction

As biology enters an era where the genomes of several organisms have been completely sequenced, the next great challenge is determining gene regulatory networks. Every gene has one or more *activators*, biochemical signals which are necessary to start transcription of the gene. Without the presence of the activator, only low level expression of the given gene can occur. Genes also have *inhibitors*, biochemical signals which prevent the expression of a particular gene even in the presence of an appropriate activator. Only a small number of genes function as activators or inhibitors, but identifying them is an important and difficult problem. A *gene regulatory network* defines the complicated structure of gene products which activate/inhibit other gene products.

Identifying gene regulatory networks from experimental data is now an area of extremely active research. New experimental technologies in molecular biology (particularly oligonucleotide arrays [11] and micro arrays) now make it possible to quickly obtain vast amounts of data on gene expression in a particular organism under particular conditions. For example, Cho, et.al [3] recently published a 17-point time series data set measuring the expression level of each of 6601 different genes for the yeast *Saccharomyces cerevisiae*, obtained using an Affymetrix hybridization array. Wen, et.al [16] has generated 9-point times series for the expression levels using RT-PCR of each of 112 genes involved in the rat nervous system development. Associating functions to genes based on this huge amount of data is an important and challenging problem.

*Department of Genetics, Harvard Medical School, 200 Longwood Avenue, Boston, MA 02115, tchen@salt2.med.harvard.edu

†Department of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794-4400, vlfilkov@cs.sunysb.edu

‡Cooresponding author. Department of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794-4400, skiena@cs.sunysb.edu. This work was partially supported by NSF Grant CCR-9625669 and ONR award 431-0857A.

In this paper, we propose a methodology for making sense of large, multiple time-series data sets arising in expression analysis, and evaluate it both theoretically and through a case study. First, we build a graph representing all putative activation/inhibition relationships by analyzing the expression profiles for all pairs of genes. Second, we prune this graph by solving a combinatorial optimization problem to identify a small set of interesting candidate regulatory elements. We do not assert that we identify "the" regulatory network as a result of this computation. However, we believe that our approach quickly enables biologists to identify and visualize interesting features from raw expression array data sets.

We have implemented our methodology and applied it to the analysis of the *Saccharomyces cerevisiae* data set. In this paper, we report on our implementation and the results of our data analysis.

The problem of inducing gene regulation networks has recently come to the computational biology community. Initial attempts at modeling gene expression abd programs to induce regulatory networks from data include [2, 10, 13]. To take fullest advantage of laboratory experiments that can be performed in which a given set of genes can be explicitly expressed or repressed, and the consequences of these genes on expression biologically determined, Akutsu, et.al. [1] considers the problem of designing a minimum-size series of experiments guaranteed to result in the identification of the correct regulatory network.

The candidate regulatory network proposed by our system depends upon the specific optimization criteria employed in the second phase of our procedure, although our experiments suggest that the optimal network is surprisingly robust to changes in the objective function. We use a simulated annealing-based optimizer to provide the maximum flexibility in our prototype system. However, a natural objective criteria suggests an interesting combinatorial problem. For this particular model, the *maximum gene regulation problem*, we present several algorithmic and complexity results, including:

- We show that the maximum gene regulatory problem is NP-complete, even for DAGs with highly restricted vertex degrees.

- We provide two different approximation algorithms for this problem. The first, based on a simple but counter-intuitive randomized assignment, achieves an approximation ratio of $((\sqrt{5}-1)/2)^2 \approx 0.38$. The second, based on linear programming relaxation, yields a 1/2-factor approximation.

- We provide an $O(n^{3/2})$ exact algorithm for the special case of networks of outdegree ≤ 2.

Our paper is organized as follows. We introduce our model in Section 2, and report on our implementation and case study in Section 3. In Section 4, we present our algorithmic results for the maximum gene-regulation problem.

2 A Model for Analyzing Expression Data

Our approach begins with a data set monitoring the expression level of each gene as a function of time. Such data sets are now becoming available. J. DeRisi et.al [4] recently made public a seven-point time series dataset for each gene in *Saccharomyces cerevisiae*. An even better dataset, with 17 sample points per gene, has been produced by Cho, et.al [3] and is the focus of our experimental work.

By analyzing the expression data, we can determine whether gene a is a candidate activator or inhibitor of gene b. If the peak of a's growth curve occurs before the leading edge of b's growth curve, then a is a candidate activator of b; if the peak of a's growth curve occurs before the trailing edge of b's growth curve, then a is a candidate inhibitor of b. Most of these relations are spurious, so we seek to identify (or at least suggest) the real regulatory network from the data.

The result of this analysis is an edge-labeled directed graph, where the vertices of the graph correspond to specific genes, and an activator-labeled (inhibitor-labeled) arc (a, b) means that a is a candidate activator (inhibitor) of b.

We seek to delete excess edges from this graph so as to maximize the number of vertices with at least one activator and one inhibitor edge into it, subject to the constraint that the remaining edges from each vertex are either all activators or all inhibitors. Thus no single gene functions in the roles of both activator and inhibitor. This is a reasonable simplifying assumption biologically since activation and inhibition operate by different mechanisms, although exceptions no doubt exist. To summarize, we define the following *maximum gene regulation problem*:

> Given a directed graph with (A/I) labeled edges, assign each vertex either an A or I label so as to maximize the number of vertices with both input A and I labels, after deleting all whose label differs from its parent vertex.

We will use a variant of this optimization function in our experiments, which also seeks to minimize the number of regulatory (A or I) elements. as reported in Section 3.5. Theoretical results for the original optimization function are presented in Section 4.

3 Implementation and Evaluation

To evaluate the effectiveness of our methodology on real data, we experimented with the *Saccharomyces cerevisiae* data set of [3] that measures the expression level of each of the 6601 ORFs of *Saccharomyces cerevisiae* at 17 points, sampled every ten minutes during roughly two complete cell division cycles.

We sought to extract enough information from this data to build a candidate activation/inhibition relationship graph, and analyze this graph to suggest possible regulatory elements. Dealing with the inherent errors in such data sets requires us to first employ certain signal processing procedures before we can effectively analyze the expression data.

3.1 Pre-Filtering

Our first processing step was to filter away ORFs which did not appear to contribute to regulation, either (1) because their absolute expression levels were below a detection threshold where fluctuations were more likely noise than signal, or (2) the gene was expressed, but showed so little variation over time as to imply that it is probably inactive or not involved in regulation.

For these reasons we employed the following filtering criteria:

- *Absolute expression* – Only accept ORFs that had an expression of > 200 in at least one of the 17 data points.

- *Relative expression* – Only accept ORFs whose maximum and average expression levels satisfied $(MAX - AVG)/AVG > 0.1$.

After employing these filters, 3131 ORFs remained in our data set.

3.2 Clustering

Many of the remaining time series expressions had such closely aligned signatures that we deemed them as having identical regulatory impact. For this reason, we decided to cluster them so that any two ORFs in the same cluster were highly similar to each other. To compare ORF expression profiles the correlation coefficient of the profiles as a similarity measure. We set a lower bound of 0.85 below which we didn't allow two different clusters to merge.

We used the well-known average linkage method of clustering [8], where each of the clusters was identified by an average, or consensus, function, at any point during the execution of the algorithm. This consensus was obtained as an average of all the ORFs that belonged to that cluster. Once the clusters were determined, the average expression functions became the consensus function we used to represent the cluster for the rest of the project.

We used the average linkage method because:

- Single linkage methods gave rise to long chains of ORFs, in which points at the opposite ends were not necessarily strongly correlated;

- Complete linkage methods were too restrictive in that they yielded many small clusters, which still resembled each other very much.

To minimize the effect of chaining, we set an upper limit on the number of ORFs per cluster. Forcibly breaking a large cluster into two clusters had no adverse implications for our method, since these two clusters would presumably show the same regulatory behavior.

In the end we got 308 well-separated clusters, examples shown in Figure 1. We note that [3] reports on a cluster analysis of the 416 cycle-dependent periodic genes in this data set.

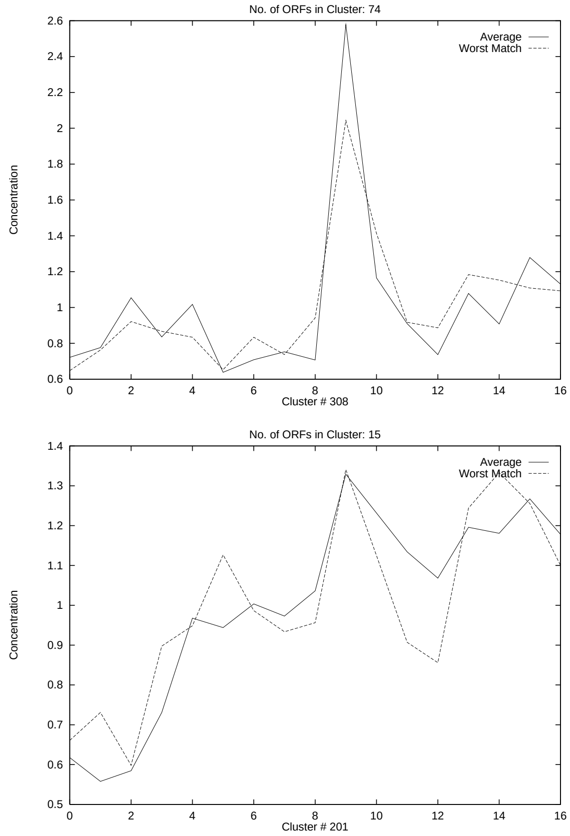

Figure 1: Large, but representative ORF clusters in *Saccharomyces cerevisiae.*

3.3 Smoothing

To facilitate the identification of candidate activation / inhibition relationships between consensus functions, we simplified each time series in the following way. In our filtered, consensus ORF expression functions, peaks describe potentially meaningful expression events, so we partition each expression profile into peaks. Each peak is described by its start point, maximum point and end point, and interpolated linearly in our simplified representation.

We used the following criteria to find peaks:

- Any data point with greater than average expression is a part of a peak; and
- All consecutive points with greater than average expression, lying between two points with less than average expression belong to the same peak.

We decided to use these smoothed profiles instead of the raw data itself because it was apparent that experimental conditions created non-monotonicities in the raw data. We also didn't use a standard convolution smoothing, based on Fourier transforms because (1) our time series did not contain sufficient data points to benefit from such filtering, (2) the output of the Fourier convolution would have had to be simplified again, at which point it would have became very similar to our triangle model, and (3) our algorithm was much simpler to implement, and faster to execute.

This smoothing technique gave us good results, as shown in Figure 2. This is because most of the peaks in the original data were narrow (at most 4 time units wide), and usually only 2 time units wide. For some of the wider peaks (wider than 4 time units) this method gave arguable, but not unreasonable results.

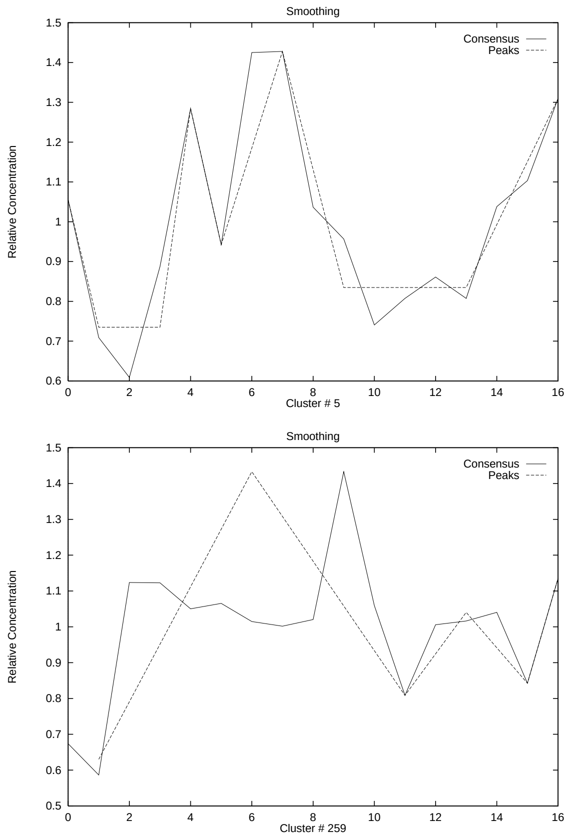

Figure 2: Good and not-as-good examples of cluster smoothing in *Saccharomyces cerevisiae.*

3.4 Activation-Inhibition Scores

Once the peaks have been identified, they define the leading and trailing edges we will use to determine the activation/inhibition grades. We used the following objective function to measure how strongly one peak activates or inhibits another. Intuitively, peak A is a good candidate to activate peak B if the "leading edge" of peak A appears "slightly before" the "leading edge" of peak B. Similarly, peak A is a good candidate to inhibit peak B if the "leading edge" of A is after the "leading edge" of B, and "close enough" to the "trailing edge" of B.

More formally, we measure the activation grade $G_a(A, B)$ as:

$$G_a(A, B) = C_3/(e^{(D_1+D_2/2)})$$

where $D_1 = B_{start} - A_{start} + 1$ and $D_2 = |B_{max} - A_{max}|$. The exponential decay ensures that this grade falls off very rapidly with distance, and the function gives more weight to the distance between the start points of the leading edges (D_1) than to the distance between their maximal. A similar grade is given for inhibition:

$$G_i(A, B) = C_4/(e^{(D_1+D_2/2)})$$

where $D_1 = B_{max} - A_{start} + 1$ and $D_2 = B_{end} - A_{max}$. Finally the total grade is obtained as follows by aligning the peaks between two profiles, awarding all matched peaks A, B the score $G_a(A, B)$ or $G_i(A, B)$, and paying a penalty of $C_5((w - 1) \times h)$ for any unmatched peak, where w is the width and h is the height of the peak, thus making the penalty proportional to the area of the peak. The constants we used were $C_3 = C_4 = 1.0$ and $C_5 = 0.1$. Examples of good activator and inhibitor candidates appear in Figure 3.

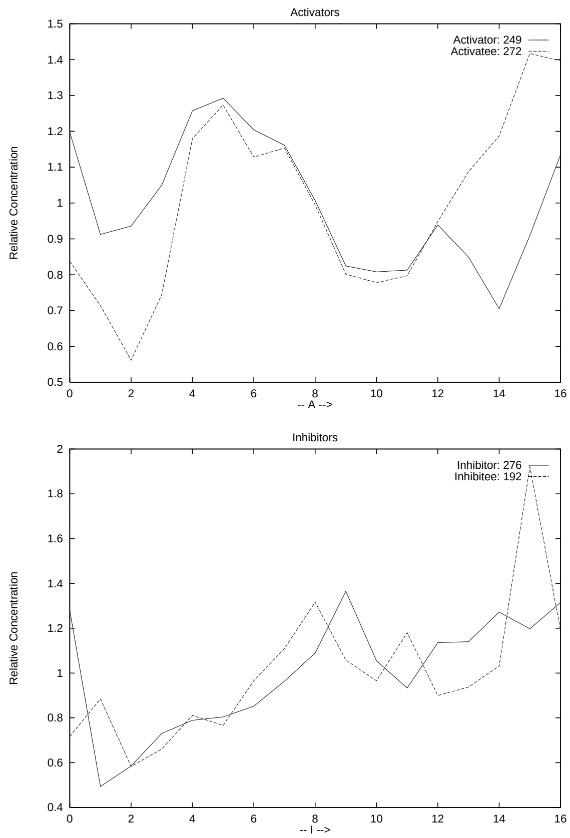

Figure 3: Examples of good candidate activator (on top) and inhibitor (on bottom) pairs.

3.5 Optimization

We used a simulated annealing-based optimizer to refine our candidate graph. The optimizer takes as input a graph in which the edges have a label (A or I) with a numerical weight measuring the strength of the given labeling. The optimizer outputs a suggested labeling of the vertices in the graph, which maximizes a given optimization function. In the experiments reported below, we seek to maximize $f(G)$, where

$$\begin{aligned} f(g) &= -C_1(count(A) + count(I)) + \\ &\sum_{v_i \in V(G)} \max(v_i[I]) \times \max(v_i[A]) \end{aligned}$$

where for each vertex we get credit for the maximal inhibiting and activating in-edge and we pay penalty proportional to the total number of vertices that are not labeled either A nor I. This function reflects our desire to have small number of regulatory elements which in turn activate a lot of things using strong activation edges. Note that the optimization function used in Section 4 does not capture this notion of minimizing the number of candidate regulatory elements.

The state transition mechanism we employ randomly changes the label of a randomly chosen vertex, and updates the cost if there is any change.

3.6 Results

To identify the constants which identify the most interesting candidate networks, we experimented 10 different values for the penalty constant C, ranging from 0.1 to 4.0 in equal intervals on each of three different cutoff values weakest edges. The flat regions in the curves of activating/inhibiting vertices and regulated vertices versus C, presented in Figure 4, demonstrate that our network is fairly robust in the face of difference choices of the critical constants.

To derive a single consensus network from a series of 100 simulated annealing runs, Figure 5 presents size of the induced subgraph of the network whose vertices were labeled consistent in at least p% of the runs. Again, the long flat regions shows that our network is relatively robust in the face of different parameters.

From these plots, we selected the parameters $C = 2$, cutoff $= 0.5$ and $p = 95\%$ as the basis for our final candidate network, shown in Figure 11. This network contains 7 proposed activators and 8 proposed inhibitors, and such interesting features as activator/inhibitor pair (clusters 107 and 170), which between them fifteen other clusters. Prof. James Konopka, a yeast specialist in the Dept. of Microbiology at Stony Brook, reviewed our network and observed several potentially interesting features, including (1) a cell division cycle regulator (CDC11 YJR083C) in activator cluster 266, (2) genes involved in DNA replication in inhibitor cluster 93, and (3) several genes involved with amino acid synthesis in regulator clusters 170 and 254. The identity of all ORFs in each regulatory cluster in included in the appendix.

4 Algorithmic and Complexity Results

In this section, we consider theoretical issues in identifying the optimal regulatory network when we seek to

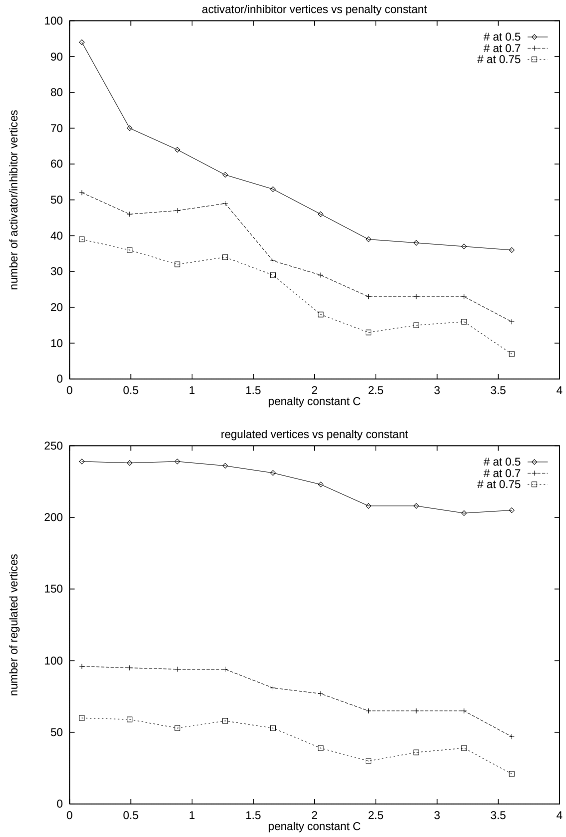

Figure 4: The effect of edge threshold and penalty constants on the proposed regulatory network.

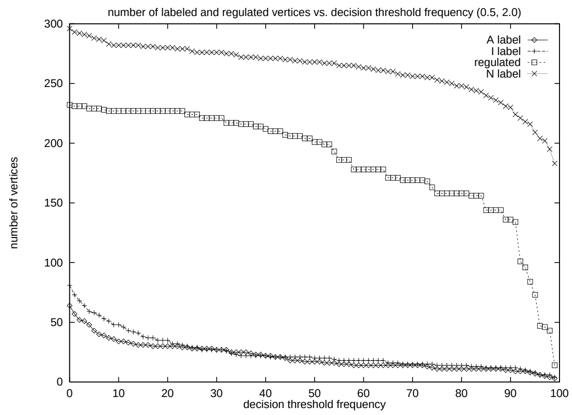

Figure 5: The effect of decision threshold on the number of labeled and regulated vertices.

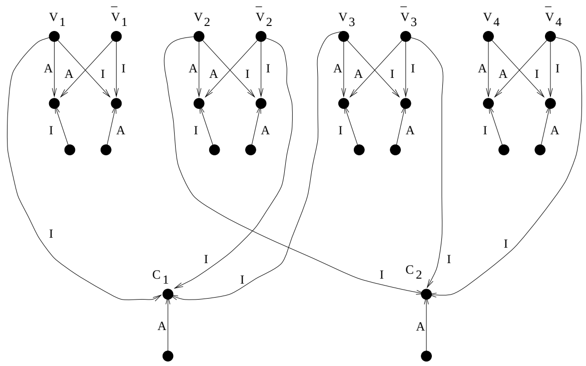

Figure 6: Reducing the formula $\{\{v_1, \overline{v}_2, v_3\}, \{v_2, \overline{v}_3, \overline{v}_4\}\}$ to an instance of maximum gene regulation.

maximize the number of genes which are both activated and inhibited, subject to the constraint that each gene may be an activator or an inhibitor, but not both. In Section 4.1, we establish that the problem is hard even for very restricted networks. Approximation results are provided in Section 4.2. Finally, exact algorithms for interesting special cases are discussed in Section 4.3.

4.1 Hardness of Gene Regulation

It is not surprising that the maximum gene regulation problem is hard, but it perhaps surprising that its hardness does not require feedback edges, as is the case for the stable gene regulatory problem of [1].

Theorem 1 *Gene regulation is NP-complete, even for directed-acyclic graphs of constant in/out-degree.*

Proof: Clearly gene regulation is in NP – guess a vertex-label assignment, delete all mislabeled edges from the graph, and verify that the requisite number of vertices are *satisfied* – meaning that they have both an activator and inhibitor edge in the final subgraph.

To show hardness, we use a reduction from 3-SAT. Each boolean variable V_i will be represented by four vertices $(V_i, \overline{V}_i, V_{iA}, V_{iI})$ with the following edges: (V_i, V_{iA}), $(\overline{V}_i, V_{iA})$ labeled as activators, and (V_i, V_{iI}), $(\overline{V}_i, V_{iI})$ labeled as inhibitors. Further V_{iA} has an inhibitor input, while V_{iI} has an activator input. The only way to satisfy both V_{iA} and V_{iI} is to assign vertices V_i and $\overline{V}_i$ to be different types (activator/inhibitor), thus corresponding to the truth assignments true and false.

Each clause will be represented by a two-vertex gadget. The first vertex of the clause gadget will have 0 indegree, and one outgoing edge (labeled activator) to the second vertex of the gadget. This second clause vertex C_i will have incoming edges (labeled inhibitor) from each literal vertex in the defining clause. See Figure 6 for an example. It can be easily verified that all the vertices with nonzero indegree can be satisfied if and only if the original boolean formula is satisfiable.

Since 3-SAT remains hard even if no literal appears in more than five clauses [5], gene regulation remains hard if each vertex has outdegree at most 7 and indegree at most 4. ∎

4.2 Approximability

In this section, we consider two classes of randomized approximation algorithms for the problem of maximum gene regulation. In Section 4.2.1, we obtain a factor ≈ 0.38 approximation algorithm, which is improved to a factor 0.5 in Section 4.2.2. We note that the former heuristic is perhaps of more practical interest, because it can be implemented easily and efficiently, thus capable of handling the graphs associated with the genomes of yeast and higher organisms.

4.2.1 Biased Random Vertex Assignment

Observe that assigning the label of each vertex to be A or I uniformly at random yields a factor 1/4 approximation. Each vertex that can be expressed must have at least one A and I edge into it, each of which will be labeled appropriately with probability 1/2. By linearity of expectation, this means we expected to satisfy at least 1/4 of optimal.

Note that this strategy does not even look at the outgoing edges of each vertex to whether there are in fact any A or I edges. A seemingly smarter idea, to weight the probability of labeling a vertex A or I according to the fraction of labeled outgoing edges, doesn't lead to a constant factor approximation. Consider a complete directed graph on n vertices, with a directed cycle of n I edges, and the rest of the edges labeled A. This randomized algorithm will pick roughly one vertex to be labeled I, and only activate one gene. However, the optimal strategy would be to select all but one vertex to be I, thus activating $n-1$ genes.

Consider the following randomized vertex labeling algorithm. For every vertex v, if the number of activator (inhibitor) outedges is greater than the number of inhibitor (activator) outedges, assign label activator (inhibitor) to v with probability p and label inhibitor (activator) to v with probability $1-p$. Note that this assignment remains biased even if the number of activator outedges equals the number of inedges. We fix a preferred label and stick to it for all balanced vertices of the graph.

This algorithm partitions the edges of G into two classes according the probability of selection, where majority class E_1 edges were selected with probability p and minority class E_2 edges were selected probability $1-p$. Denote the number of edges in E_1 and E_2 by e_1 and e_2, respectively. This assignment scheme implies that $e_1 \geq e_2$.

Lemma 2 *For any indegree ≤ 2 network G, the above assignment approximates the maximum gene regulation to within a factor of p^2 for any $1/2 \leq p \leq 2/3$.*

Proof: The vertices whose inputs are of same label will never be satisfied and thus are excluded. Then we group the rest set of vertices with exactly two inputs into three classes according to whether its input edges are (1) two members of E_1, (2) one member of E_1 and one of E_2, or (3) two members of E_2. Let v_1, v_2, and v_3 denote, respectively, the number of vertices for these three classes, and $v = v_1+v_2+v_3$. The expected number of satisfied vertices $Apx(G)$ for G under this scheme is:

$$\begin{aligned} Apx(G) &= p^2 v_1 + p(1-p)v_2 + (1-p)^2 v_3 \\ &= p^2 v + (p-2p^2)v_2 + (1-2p)v_3 \end{aligned}$$

Observe that $e_2 = v_2 + 2v_3$, since each vertex in v_2 is incident one E_2 edge and each each vertex in v_3 on two E_2 edges. A simple argument from linear programming gives the following inequalities for the last two terms in $Apx(G)$:

$$(p-2p^2)e_2 \leq (p-2p^2)v_2 + (1-2p)v_3 \leq (\frac{1}{2} - p)e_2$$

As each vertex has only two input edges, $v = (e_1 + e_2)/2$ and so

$$\begin{aligned} Apx(G) &\geq p^2 v + (p-2p^2)e_2 \\ &\geq p^2 e_1/2 + (p - 3p^2/2)e_2 \end{aligned}$$

For $1/2 \leq p \leq 2/3$ then $p - 3p^2/2 \geq 0$ so $Apx(G) \geq p^2 e_1/2$.

Recall that $e_1 \geq e_2$, and so the optimal assignment can validate at most e_1 edges and thus at most $e_1/2$ vertices. Thus the ratio of $Apx(G)/Opt \geq (p^2 e_1/2)/e_1/2 = p^2$, giving the result. ∎

Corollary 1 *For any ≤ 2 indegree network G, the above assignment yields a 4/9-approximation to maximum gene regulation, when $p = 2/3$.*

Corollary 1 is not the best result possible – we note that a factor 1/2 approximation algorithm for indegree 2 networks follows from Trevisan's [14] factor 2^{1-k} approximation of maximum k conjunctive satisfiability. However, Lemma 2 can be used to generalize our assignment to arbitrary degree gene regulatory networks. Since any vertex without both acceptor and inhibitor-labeled edges can never be satisfied, its input edges can be deleted from G without effect.

Theorem 3 *Maximum gene regulation of G can be approximated to a factor of $((\sqrt{5}-1)/2)^2 \approx 0.38$ times optimal, with high probability.*

Proof: Our probability assignment scheme has two steps. First, we construct an indegree-2 network G_1 as follows. All indegree-2 nodes from G will appear in G_1. G_1 will also contain two super source vertices, one for activators (s_a) and one for inhibitors (s_i) are added, shown as black nodes in Figure 7. Any vertex t from G which is the sink for multiple activator (inhibitor) edges and one inhibitor (activator) edge will have these activator (inhibitor) edges replaced in G_1 by one edge from s_a (s_i). See Figure 7. Thus every vertex in G_1 has at most two inputs.

By Lemma 2, setting $p = (\sqrt{5}-1)/2$ yields an approximation of $((\sqrt{5}-1)/2)^2$ to the optimal in G_1. For each of the vertices in G_1, we make the same probability assignment to the corresponding vertices in G. Now we assign all unassigned vertices to be activators with probability p.

We claim that this assignment for G has an expected approximation factor of p^2. Observe that any vertex with ≥ 2 activator inputs and ≥ 2 inhibitor inputs have at least a $(1-p^2)^2 > p^2$ probability of being satisfied, for $p = (\sqrt{5}-1)/2$. Any vertex from G_1 was satisfied

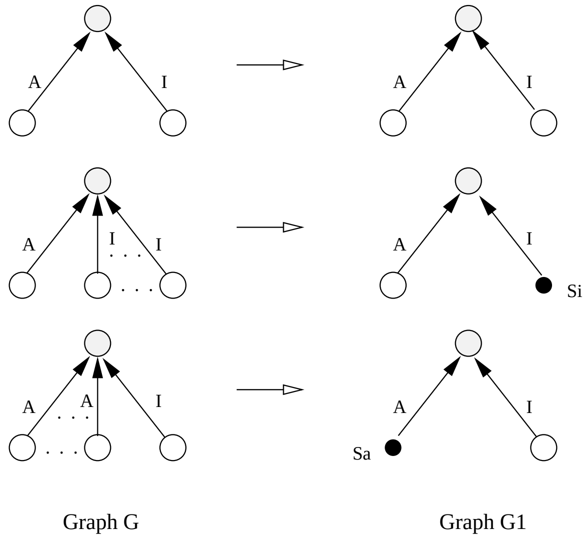

Figure 7: Building an indegree-2 network G_1 from G by adding super source vertices s_a and s_i.

in G_1 with probability $\geq p^2$, and so it is in G. Each remaining vertices in G has one input activator-labeled (inhibitor-labeled)) and at least two inputs labeled inhibitor (activator). They are more likely satisfied in G than G_1, because the two super source were correctly labeled in G_1 with probability p, while the probability of at least one of the ≥ 2 majority edges being correctly labeled is at least $1-p^2$, which equals p for $p = (\sqrt{5}-1)/2$. ∎

4.2.2 An Improved Approximation Algorithm

The best approximation algorithms known for many logic optimization problems follow from solving a linear or semidefinite programming relaxation, and then rounding the optimized solutions into integer solutions for the original problem. Goemans and Williamson [6] first used positive linear programming relaxation to give a 3/4-approximation algorithm for maximum satisfiability, and later [7] improved the factor to 0.758 by using semidefinite programming relaxation. Trevisan [15] gives a 2^{1-k}-approximation algorithm for MAX k CONJ SAT problem. Here we show that maximum gene regulation can be approximated to a 1/2 factor using positive linear programming relaxation.

Let C_j denote the boolean formula associated with vertex v_j, and let C_j^+ and C_j^- denote the set of activators and inhibitors respectively Thus

$$C_j = (\bigvee_{i \in C_j^+} v_i) \wedge (\bigvee_{i \in C_j^-} \neg v_i)$$

Clearly v_j is satisfied if and only if C_j is satisfied. Consider the following integer programming formulation of the maximum gene regulation problem.

$$\text{LP : Maximize } \sum_j z_j$$

$$\text{Subject to } \begin{cases} z_j \leq \sum_{i \in C_j^+} x_i & \text{for all j} \\ z_j \leq \sum_{i \in C_j^-} (1-x_i) & \text{for all j} \\ 0 \leq x_i \leq 1 & \text{for all i} \\ 0 \leq z_j \leq 1 & \text{for all j} \end{cases}$$

If we associate x_i with v_i and z_j with C_j, the linear programming solution of this LP provides an upper bound to the solution of the maximum gene regulation problem. However, a feasible solution $(\mathbf{x}, \mathbf{z})$ may not be integral.

Lemma 4 *Let ($\mathbf{x}$, $\mathbf{z}$) be a feasible solution for this LP. Randomly assign v_i to be 0 or 1 according to the value of x_i:*

$$\mathbf{Pr}[v_i = 1] = \frac{1}{4} + \frac{x_i}{2}$$

Then

$$\begin{aligned} \mathbf{L_j} &= \mathbf{Pr}[\bigvee_{i \in C_j^+} v_i = 1] \geq \frac{1}{4} + \frac{z_j}{2} \\ \mathbf{R_j} &= \mathbf{Pr}[\bigvee_{i \in C_j^-} v_i = 0] \geq \frac{1}{4} + \frac{z_j}{2} \end{aligned}$$

Proof: Let k be the size of C_j^+, thus

$$\begin{aligned} L_j &= 1 - \prod_{i \in C_j^+} (1 - (\tfrac{1}{4} + \tfrac{x_i}{2})) \\ &= 1 - \prod_{i \in C_j^+} (\tfrac{3}{4} - \tfrac{x_i}{2}) \\ &\geq 1 - (\frac{\sum_{i \in C_j^+} (\frac{3}{4} - \frac{x_i}{2})}{k})^k & \text{for } (\tfrac{y_1 + \ldots + y_k}{k})^k \geq y_1 \ldots y_k \\ &= 1 - (\tfrac{3}{4} - \frac{\sum_{i \in C_j^+} x_i}{2k})^k \\ &\geq 1 - (\tfrac{3}{4} - \tfrac{z_j}{2k})^k & \text{for } z_j \leq \textstyle\sum_{i \in C_j^+} x_i \text{ in LP} \end{aligned}$$

Let $Y = 1 - (\frac{3}{4} - \frac{z_j}{2k})^k$ and the first order derivative of Y is

$$\frac{\mathbf{d}Y}{\mathbf{d}k} = -(\frac{3}{4} - \frac{z_j}{2k})^k \times \ln(\frac{3}{4} - \frac{z_j}{2k}) \times k \times \frac{1}{\frac{3}{4} - \frac{z_j}{2k}} \times \frac{z_j}{2k^2} \geq 0$$

when $k \geq 1$ and $0 \leq z_j \leq 1$, which means Y monotonically increases in the range of $k \in [1, +\infty)$. Thus

$$Y|_{k \geq 1} \geq Y|_{k=1} = \frac{1}{4} + \frac{z_j}{2}$$

and it holds when k is a positive integer (in L_j) and so

$$L_j \geq Y|_{k=1} = \frac{1}{4} + \frac{z_j}{2}$$

The bound for R_j follows similarly. ∎

For any feasible solution $(\mathbf{x}, \mathbf{z})$, this rounding scheme satisfies C_j with expected value:

$$\begin{aligned} E(C_j) &= \mathbf{Pr}[\bigvee_{i \in C_j^+} v_i = 1] \times \mathbf{Pr}[\bigvee_{i \in C_j^-} v_i = 0] \\ &= \mathbf{L_j R_j} \\ &\geq (\frac{1}{4} + \frac{z_j}{2})^2 \geq \frac{z_j}{2} \end{aligned}$$

since

$$(\frac{1}{4}+\frac{z_j}{2})^2-\frac{z_j}{2}=(\frac{1}{4}-\frac{z_j}{2})^2\geq 0$$

Thus

$$E(\mathbf{S})=\sum_j E(C_j)\geq\sum_j\frac{z_j}{2}=\frac{1}{2}\sum_j z_j$$

Since this result holds for any feasible solution, including the optimal one:

Theorem 5 *This randomized rounding scheme yields an $\frac{1}{2}$-approximation to the maximum gene regulation problem.*

4.3 Restricted Degree Networks

In light of our hardness results, we consider the special cases of networks of restricted degrees. We give an efficient algorithm for case where each vertex has outdegree ≤ 2. Conversely, we show that the problem is hard even when each vertex has indegree ≤ 2.

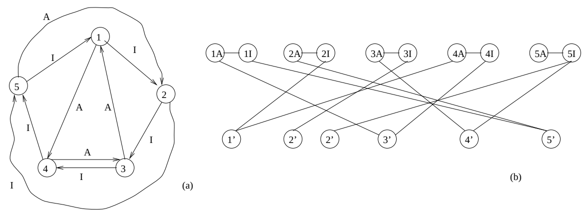

Figure 8: (a) a 2-input, 2-output network and (b) the resulting matching graph.

For any network G we define the following *dual graph* $D(G)$. We say a vertex v in G is *variable* if it has both outgoing activator and inhibitor edges; otherwise v is either a *fixed activator* or *fixed inhibitor* vertex. Each variable vertex v_i of G will define three vertices in $D(G)$; v_{iA}, v_{iI}, and v_i'. Each fixed vertex v_i will define vertices v_{iA} and v_{iI} plus the outdegree of v_i many vertices each labeled v_i'. Each activator edge (v_i, v_j) of G defines an edge (v_i', v_{jA}), while inhibitor edges (v_i, v_j) of G defines an edge (v_i', v_{jI}). In the case of fixed vertices, care is taken to use each copy of v_i' only once. Finally, $D(G)$ will contain an edge (v_{iA}, v_{iI}) for each vertex v_i of G, whether it is fixed or variable. This construction is illustrated in Figure 8.

Lemma 6 *For any maximum output degree ≤ 2 network G, the maximum number of satisfiable vertices in G equals the cardinality of the maximum matching in $D(G)$.*

Proof: First, we show how to construct a cardinality $x+n$ matching in $D(G)$ for any vertex labeling of G which satisfies x of n genes/vertices. For each satisfied vertex v_j, the vertex labeling defines at least one pair of activator and inhibitor vertices v_a' and v_i', and so the matching contains edges (v_a, v_{jA}) and (v_i, v_{jI}). For each unsatisfied vertex v_k, the matching will contain the edge (v_{kA}, v_{kI}). This defines a matching since any variable vertex v_j will be used at most once, and there are sufficient copies of any fixed vertex to be used as needed.

Conversely, suppose we are given a maximum matching M of size $x+n$ in $D(G)$. If any pair of vertices v_{kA} and v_{kI} for some $1\leq k\leq n$ are not both used in M, we can replace the matching edge adjacent to one of them with the edge (v_{kA}, v_{kI}) without changing the size of the matching. Now each vertex v_i' will take the label of the vertex to which it is adjacent in the matching. The vertices which are satisfied by this labeling are exactly the ones for which the edge (v_{kA}, v_{kI}) do not appear in the matching, of which there must be x of in a matching of size $x+m$. ∎

Theorem 7 *Maximum gene regulation can be solved in $O(n^{3/2})$ time for n-vertex networks of outdegree ≤ 2. Further, it can be solved in linear time when each gene has indegree and outdegree of at most two.*

Proof: By Lemma 6, we have reduced this problem to maximum cardinality matching in $D(G)$. Micali and Vazirani [12] find maximum matchings in $O(\sqrt{n}m)$ time. The result follows since dual graph contains $O(n)$ edges.

For the indegree 2 case, observe that each vertex in $D(G)$ must have degree at most two when each vertex of G has indegree and outdegree of at most two. Therefore, the connected components of $D(G)$ consist only of paths and cycles, and so the matching can be found via depth-first search in linear time. ∎

For networks of indegree ≤ 2, there is at most one possible vertex label assignment to satisfy any given vertex. This constraint was exploited in Theorem 7 to yield an efficient algorithm for the indegree 2, outdegree 2 case. We can exploit connections between the following logic problems to show that general indegree ≤ 2 networks are not sufficient to yield a polynomial-time algorithm.

- *Maximum 2-SAT* – Given a set of clauses each consisting of exactly two boolean literals, find the assignment which maximizes the number of satisfied clauses.
- *Minimum 2-SAT* – Given a set of clauses each consisting of exactly two boolean literals, find the assignment which minimizes the number of satisfied clauses.
- *Maximum 2-of-2-SAT* – Given a set of clauses each consisting of exactly two boolean literals, find the assignment which maximizes the number of satisfied clauses, where both literals must be true to satisfy a clause.
- *Restricted maximum 2-of-2-SAT* – Given a set of clauses each consisting of exactly two boolean literals (one of which is complemented), find the assignment which maximizes the number of satisfied clauses, where both literals must be true to satisfy a clause.

Lemma 8 *Restricted maximum 2-of-2-SAT is NP-complete.*

Proof: Maximum 2-SAT and minimum 2-SAT are well-known NP-complete problems [9]. The hardness of maximum 2-of-2-SAT follows by a simple reduction from minimum 2-SAT, where both literals of each clause (x, y) is converted into the clause $(\bar{x}, \bar{y})$. Needing both x and y to satisfy a clause yields the boolean formula (x and y) $=$ not (not x or not y). Thus maximizing the number of and terms equals minimizing the number of or terms.

The hardness of restricted maximum 2-of-2-SAT follows by a simple reduction from maximum 2-of-2-SAT, since any 2-of-2-SAT clause (x, y) is equivalent to the restricted 2-of-2-SAT clause $(x, \bar{y})$. ∎

Theorem 9 *Maximum gene regulation is NP-complete, even when restricted to networks of indegree ≤ 2.*

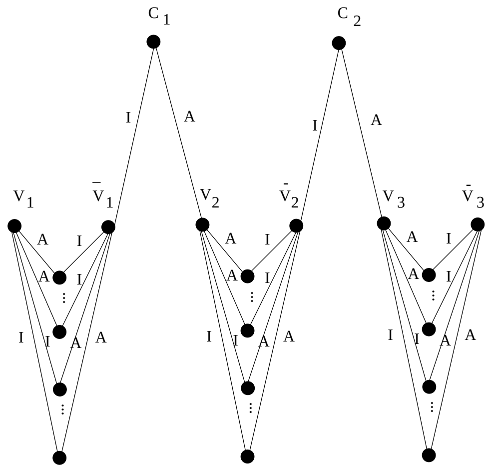

Figure 9: Reducing the boolean formula $\{\{\overline{v}_1, v_2\}, \{\overline{v}_2, v_3\}\}$ to an instance of indegree ≤ 2 maximum gene regulation.

Proof: By a reduction from restricted maximum 2-of-2-SAT. We construct a low indegree variable gadget with A edges (v_i, v_{yi1}) and $(\overline{v}_i, v_{yi2})$ and I edges (v_i, v_{yi2}) and $(\overline{v}_i, v_{yi1})$ enforces that v_i and $\overline{v}_i$ have opposite labels to satisfy one of each pair of v_{yi1} and v_{yi2}. With sufficiently many such pairs of vertices, we ensure that the maximum gene regulation conforms with a truth assignment. This construction is illustrated in Figure 9.

Each clause $C_x = (\overline{v}_i, v_j)$ is represented by a clause vertex c_x and arcs $(\overline{v}_i, c_x)$ (labeled I) and $(\overline{v}_j, c_x)$ (labeled A). Because the reduction is from restricted 2-of-2-SAT, each clause gadget will have exactly one A and one I input, and thus satisfiable iff the 2-of-2-SAT clause is. Maximizing the number of satisfied genes is equivalent to maximizing the number of satisfied clauses. ∎

Acknowledgments

We thanks Barry Cohen, James Konopka, Herb Lewis, Prashant Pradhan, and Saurabh Sethia for helpful discussions.

References

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. Identification of gene regulatory networks by strategic gene disruptions and overexpressions. In *Proc. Ninth ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 695–702, 1998.

[2] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. In *Pacific Symposium Biocomputing '99*, pages 29–40, 1999.

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

[4] J. DeRisi, V. Iyer, and P. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, Oct. 24, 1997.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[6] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.

[7] M. X. Goemans and D. P. Williamson. Improved approxiamtion algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.

[8] A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs NJ, 1988.

[9] R. Kohli, R. Krishnamurti, and P. Mirchandani. The minimum satisfiability problem. *SIAM J. Discrete Math.*, 7:275–283, 1994.

[10] S. Liang, S. Fuhrman, and R. Somogyi. Reveal: a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium Biocomputing '98*, pages 18–29, 1998.

[11] D. Lockhart, H. Dong, M. Byrne, M. Follettie, M. Gallo, M. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. Brown. Expression monitoring by hybrdiziation to high-density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, December 1996.

[12] S. Micali and V. Vazirani. An $O(\sqrt{|V|}|e|)$ algorithm for finding maximum matchings in general graphs. In *Proc. 21st. Symp. Foundations of Computing*, pages 17–27, 1980.

[13] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In *Pacific Symposium Biocomputing '98*, pages 77–87, 1998.

[14] L. Trevisan. Positive linear programming, parallel approximation, and pcps. In *Proc. 4th European Symp. Algorithms*, volume 1136, pages 62–75, 1996.

[15] L. Trevisan. Approximating satisfiable satisfiability problems. In *Proc. 5th European Symp. Algorithms*, volume 1284, pages 472–485, 1997.

[16] X. Wen, S. Fuhrman, G. Michaels, D. Carr, S. Smith, J. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl. Acad. Sci.*, 95:334–339, 1998.

Appendix: Activator and Inhibitor Clusters

Below, we list the orfs in each cluster corresponding to proposed regulatory elements. The ORF identification names are the same used in [3]. Plots of the activator/inhibitor pair of clusters 170 and 107 appear in Figure 10.

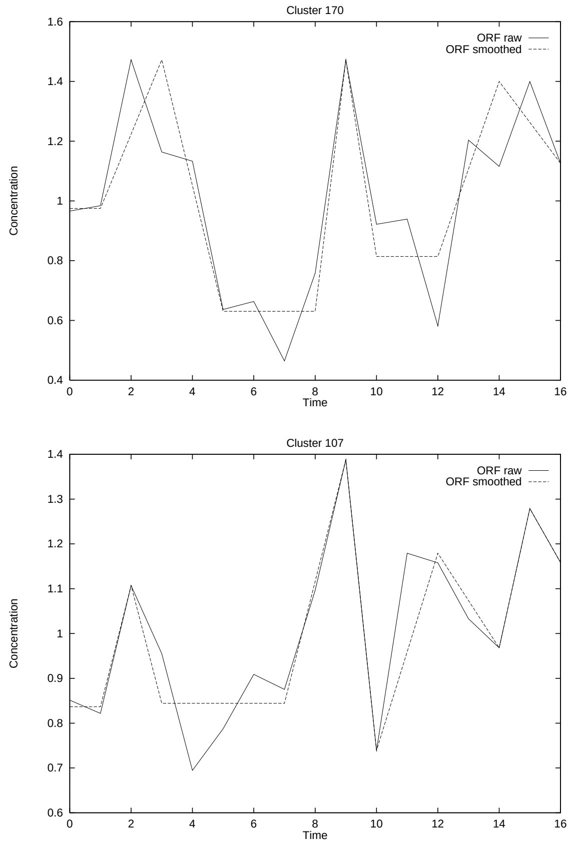

Figure 10: Expression profiles of the activator-inhibitor pair of clusters 170 and 107.

- *cluster 12 – Activator* – YCL047c/ YBR024w/SCO2
- *cluster 30 – Inhibitor* – YDR237W/ YBR151w/ YDL078C/MDH3
- *cluster 67 – Activator* – YGR282C/BGL2 YER031c/YPT31 YGL076C/RPL6A_ex1
- *cluster 93 – Inhibitor* – YKL112W/ABF1 YJR060W/CBF1

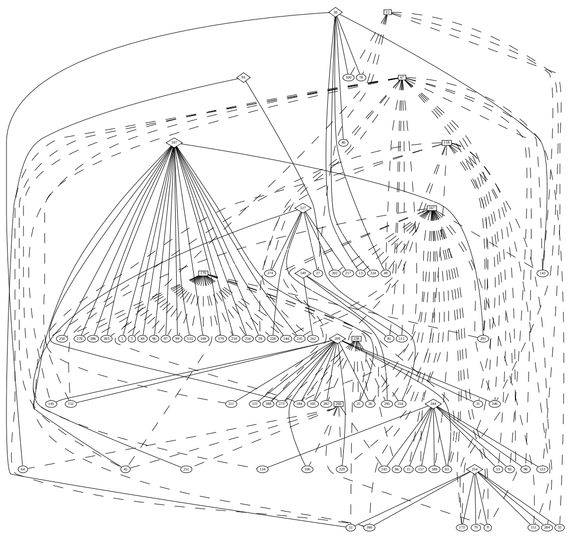

Figure 11: Candidate regulatory network for $C = 2$, cutoff $= 0.5$ and $p = 95\%$, with candidate activators in boxes and candidate inhibitors in diamonds.

- *cluster 107 – Inhibitor* – YJL143W/TIM17 YKL207W/
- *cluster 118 – Activator* – YLR058c/SHM2 YDR016c/ YDR502c/SAM2 YKL175W/
- *cluster 167 – Activator* – YMR046C/_f YMR045C/_ex1_f
- *cluster 170 – Activator* – THR3 LYSA5 LYSAM LYSA3 THR5 THR3 LYSA5 LYSAM LYSA3 LYSA5 LYSAM LYSA3 THR5 THR3 LYSA5 LYSAM LYSA3 THR5
- *cluster 209 – Inhibitor* – YOR014W/RTS1 YNL121C/TOM70
- *cluster 238 – Activator* – YOR272W/ YCR072c/
- *cluster 244 – Inhibitor* – YOR346W/ YBR086c/ YBR115c/LYS2 YEL013w/
- *cluster 254 – Inhibitor* – YPL135W/ YBL043w/ YCLX09w/ YCL025c/AGP1 YCR023C/ YDL198C/SHM1 YDR019C/GCV1 YDR068W/DOS1 YDR174W/ YDR380W/ YFR033C/QCR6 YGL186C/ YGL161C/ YHR018c/ARG4 YHR029C/ YIL116W/HIS5 YJR048W/CYC1 YJR109C/CPA2 YJR130C/ YLR093c/ YLR399C/BDF1 YMR058W/FET3 YMR062C/ YMR189W/ YNL259C/ATX1 YNL142W/MEP2 YNL129W/ YNL100W/ YOL064C/MET22 YOL058W/ARG1 YOR036W/PEP12 YOR130C/ARG11 YOR202W/HIS3 YOR203W/ YOR311C/ YPL250C/
- *cluster 266 – Activator* – YPL032C/ YDL056W/MBP1 YJL092W/HPR5 YJR076C/CDC11 YJR083C/ YJR112W/NNF1 YKL052C/ YKL049C/CSE4 YKR010C/ YLL032c/ YLR045c/STU2 YMR003W/ YMR215W/ YNL238W/KEX2 YNR009W/ YNR028W/ YOR073W/ YOR372C/