Finding regulatory modules through large-scale gene-expression data analysis

M. Kloster1,2, C. Tang2,3,* and N.S. Wingreen2,4

1Department of Physics, Princeton University, Princeton, NJ 08544, USA, 2NEC Laboratories America, Inc., Princeton, NJ 08540, USA, 3Center for Theoretical Biology, Peking University, Beijing 100871, China and 4Department of Molecular Biology, Princeton University, Princeton, NJ 08544, USA

Received on May 12, 2004; revised on September 27, 2004; accepted on October 11, 2004

Advance Access publication October 28, 2004

ABSTRACT

Motivation: The use of gene microchips has enabled a rapid accumulation of gene-expression data. One of the major challenges of analyzing this data is the diversity, in both size and signal strength, of the various modules in the gene regulatory networks of organisms.

Results: Based on the iterative signature algorithm [Bergmann et al., 2002; Ihmels et al., 2002], we present an algorithm—the progressive iterative signature algorithm (PISA)—that, by sequentially eliminating modules, allows unsupervised identification of both large and small regulatory modules. We applied PISA to a large set of yeast gene-expression data, and, using the Gene Ontology database as a reference, found that the algorithm is much better able to identify regulatory modules than methods based on high-throughput transcription-factor binding experiments or on comparative genomics.

Contact: tang@nec-labs.com

1 INTRODUCTION

The introduction of DNA microarray technology has made it possible to acquire vast amounts of gene-expression data, raising the issue of how best to extract information from this data. While basic clustering algorithms have been successful at finding genes that are coregulated to a large set of yeast gene-expression data, and, using the Gene Ontology database as a reference, found that the algorithm is much better able to identify regulatory modules than methods based on high-throughput transcription-factor binding experiments or on comparative genomics.

One clear conceptual limitation of ISA is that it only considers one transcription module at a time; the algorithm does not use knowledge of already identified modules to help it find new modules. ISA may find a strong module hundreds of times before it finds a given weak module, or it may be unable to find a weak module at all. A simple way to ensure that the same module is not found repeatedly is to directly subtract the module from the expression data (this approach is used in Lazzeroni and Owen, 2002). A more robust approach is to require the condition vector, i.e. the weighted condition set, of each new transcription module to be orthogonal to the condition vectors of all previously found modules. In essence, this procedure corresponds to successively removing transcription modules to reveal smaller and weaker modules. The successive removal of condition vectors is the central new feature in our approach.

(TMs), i.e. sets of coregulated genes along with the sets of conditions for which the genes are strongly coregulated. SA is well grounded in the biology of gene regulation. Typically, a single transcription factor regulates multiple genes; a TM naturally corresponds to a set of such genes and the conditions under which the transcription factor is active. The authors tested the algorithm on a large data set for the yeast Saccharomyces cerevisiae. By applying SA to various sets of genes that were known or believed to be related, they identified a large number of TMs.

Soon after, Bergmann et al. (2003) introduced the iterative signature algorithm (ISA), which uses the output of SA as the input for additional runs of SA until a fixed point is reached. By applying ISA to random input sets and varying the threshold coefficient $t_0$ (see below), the authors found almost all the TMs that had been identified using SA, as well as a number of new modules. Many of these modules proved to be in excellent agreement with existing knowledge of yeast gene regulation.

While ISA can identify many transcriptional regulons from gene-expression data, the algorithm has significant limitations. The recovered modules depend strongly on the value of a threshold coefficient $t_0$ used in the algorithm. To find all the relevant modules, this threshold must be varied by more than a factor of 2, and for high thresholds many of the modules appear to be due to noise. While the largest, strongest modules are easily identified, among the smaller, weaker modules it is a major challenge to identify the real transcriptional regulons. Weak modules can even be completely ‘absorbed’ by stronger modules.

One clear conceptual limitation of ISA is that it only considers one transcription module at a time; the algorithm does not use knowledge of already identified modules to help it find new modules. ISA may find a strong module hundreds of times before it finds a given weak module, or it may be unable to find a weak module at all. A simple way to ensure that the same module is not found repeatedly is to directly subtract the module from the expression data (this approach is used in Lazzeroni and Owen, 2002). A more robust approach is to require the condition vector, i.e. the weighted condition set, of each new transcription module to be orthogonal to the condition vectors of all previously found modules. In essence, this procedure corresponds to successively removing transcription modules to reveal smaller and weaker modules. The successive removal of condition vectors is the central new feature in our approach.
We call the modified algorithm the progressive iterative signature algorithm (PISA).

2 METHODS AND ALGORITHMS

2.1 Motivation

To a first approximation, the expression level of a gene is determined by the activity of the various transcription factors in the cell.\(^1\) If we assume that different transcription factors act multiplicatively on the expression level, i.e. additively on its logarithm (Buisemaker et al., 2001), then the relative expression levels of all the genes under a set of experimental ‘conditions’ are given by

\[
E = \sum_{j} g_{j} c_{j}^{T} + \eta, \tag{1}
\]

where \(E_{gj}\) is the logarithmic expression ratio of gene \(g\) under condition \(c\), relative to a reference condition. The ‘gene vector’ \(g_{j}\) specifies to what extent each gene is regulated by transcription factor \(t\) under each condition (specifically, each element of \(g_{j}\) is the log ratio of \(t\) activity in a particular condition relative to its reference condition); \(\eta\) indicates noise. Together, we call corresponding gene vector \(g_{j}\) and condition vector \(c_{j}\) a TM.

The assumption of multiplicative activities may be approximately true for single-celled organisms,\(^2\) but certainly does not capture the highly combinatorial regulation present in multicellular organisms. Nevertheless, Equation (1) is a useful model for the role of transcription modules in gene expression.

Ideally, for a given gene-expression data set we would like to extract the full set of gene vectors and condition vectors. The gene vectors describe the sets of genes that are coregulated at the most basic level, while the condition vectors describe how the cell responds to the different experimental conditions. Unfortunately, the decomposition of the matrix \(E\) given by Equation (1) is not unique, even in the absence of noise.

There are ways to find unique decompositions of \(E\) by requiring additional properties of the gene vectors and condition vectors. One such approach is singular value decomposition (SVD; see e.g. Alter et al., 2000), which leads to gene vectors (eigenarrays) that are orthogonal to each other, as are the condition vectors (eigengenes). However, these orthogonal properties do not match our biological expectations—different transcription factors may control substantially overlapping sets of genes, and may also be active under many of the same experimental conditions. In addition, as shown by Bergmann et al. (2003), SVD is sensitive to noise.

In order to find a biologically relevant decomposition, one should use the properties we expect the ‘real’ solution to have. For instance, each transcription factor typically controls only a small subset of the genes in a cell. Thus, we expect the gene vectors to be sparse. A reasonable goal is to find the ‘real’ solution to require that both gene vectors and condition vectors be sparse. However, the two thresholds are very different: the gene threshold is specified in terms of standard deviations of the observed gene-score distributions, and thus sets an absolute (\(s_{g}\)-dependent) limit on the fraction of genes that can be included in a module. The condition threshold, on the contrary, compares each score to the expected distribution (if the data was uncorrelated noise), thus there is no limit on the number of conditions that can be included. Indeed, few transcription modules found by ISA contain \(< 10\%\) of the conditions, and some contain \(> 80\%\).

As mentioned in Section 2.1, different TMs are often correlated. This can contribute to the hierarchical clustering by ISA: for a low gene-threshold coefficient \(s_{g}\), correlated modules may appear to be a single, large module, while at higher thresholds the individual modules are resolved (Bergmann et al., 2003; Ihmels et al., 2004). However, it may be impossible for SA/ISA to resolve correlated modules regardless of the value of \(s_{g}\). This is illustrated in Figure 1 for a synthetic data set with only two TMs: \(E = g_{1}e_{1}^{T} + g_{2}e_{2}^{T} + \eta\), where the elements of \(\eta\) are independently drawn from a normal distribution.

\(1\)Post-transcriptional regulation by specific degradation of mRNA may also be considered to be a ‘transcription factor’ effect in this context.

\(2\)Moreover, even for single-celled organisms, the ascription of one transcription module to each transcription factor is only approximate. For instance, a transcription factor may regulate some genes on the basis of its concentration only, while it may regulate others depending on its phosphorylation state.

\(3\)ISA actually uses two matrices with different normalizations (Ihmels et al., 2002).
2.3 The algorithm PISA

2.3.1 Orthogonalization Within PISA, each condition-score vector $s^C$ is required to be orthogonal to the condition-score vectors of all previously found TMs, as illustrated schematically in Figure 2. Therefore, whenever PISA finds a TM and its associated condition-score vector $s^C$, the component along $s^C$ of each gene is removed from the gene expression matrix (see Section 2.3.3). For example, in the toy model in Figures 1 and 2, one finds that PISA can easily identify both TMs: it first finds the strong module, removes its condition vector, and then the only signal left is that of the weak module.

Progressively eliminating TMs à la PISA can also improve the prospects for finding unrelated modules. The gene regulation from one module will contribute to the background noise for all unrelated modules. Therefore, eliminating large, strong modules can significantly improve the signal to noise ratio of the remaining modules. This is in contrast to the situation for SVD: the initial modules found with SVD will typically be a mixture of many real transcription modules, and removing them will not significantly improve the signal for weak modules. In PISA, the gene-score threshold ensures that only a few, typically highly correlated, TMs will be combined.

The requirement of orthogonality in PISA conflicts with the condition-score threshold as used in ISA. If we make the condition-score vector orthogonal first and then apply the threshold, the vector will no longer be orthogonal, whereas if we apply the threshold first, orthogonalization will give non-zero weight to all conditions, eliminating the noise-filtering benefit of thresholding. We have chosen to eliminate the condition-score threshold completely. In any event, conditions that in ISA would fall below the threshold will have low weight and will give only a small contribution to the noise.

This orthogonalization procedure gives good estimates for the gene vectors in Equation (1), but the resulting condition vectors are of course all orthogonal. A condition vector calculated from the final gene vector using the initial value of the gene-expression data matrix, as given in Section 2.3.6, gives a much better description of the ‘real’ TM.

2.3.2 The gene-score threshold In ISA, the gene-score threshold is $t_G \sigma_G$, where the standard deviation $\sigma_G$ is computed using the full distribution of gene scores and includes contributions both from the background and from the module of interest (Fig. 3). For large, strong modules, the module contribution may be larger than the background contribution. As a result, $\sigma_G$ is module dependent, and $t_G$ must be adjusted to prevent false positives from the background: at low thresholds, a small module would be lost among false positives; while at high thresholds, it is mathematically impossible to find a large module. One can run ISA with many different threshold coefficients $t_G$ in order to find more modules than available at any single threshold, however this results in a large number of false positives.

Within PISA, we eliminate the need to use multiple gene-score thresholds by specifying the threshold relative to the background alone, which we estimate using the mean, $\langle x \rangle$, and the standard deviation, $\sigma$, of the gene scores within the shortest interval that contains at least 70% of all the gene scores. By excluding extreme gene scores in this way, we minimize the influence of the module of interest itself on the means and standard deviations of gene scores (Fig. 3). As a test, we used $\sigma$ in place of $\sigma_G$ in ISA and found it to be very large and very small modules with a single value of $t_G$.

We need to be conservative when selecting the gene-score threshold because, if PISA misidentifies a module, elimination of its condition vector can lead to errors in other modules. Therefore, the number of genes included in modules due to noise should be very low. We have used a threshold of 7.0$\sigma$, which for a Gaussian distribution corresponds to about 3.9$\sigma$. The condition-score threshold $t_G$ must be adjusted to prevent false positives from the background: at low thresholds, a small module would be lost among false positives; while at high thresholds, it is mathematically impossible to find a large module. One can run ISA with many different threshold coefficients $t_G$ in order to find more modules than available at any single threshold, however this results in a large number of false positives.

Within PISA, we eliminate the need to use multiple gene-score thresholds by specifying the threshold relative to the background alone, which we estimate using the mean, $\langle x \rangle$, and the standard deviation, $\sigma$, of the gene scores within the shortest interval that contains at least 70% of all the gene scores. By excluding extreme gene scores in this way, we minimize the influence of the module of interest itself on the means and standard deviations of gene scores (Fig. 3). As a test, we used $\sigma$ in place of $\sigma_G$ in ISA and found it to be very large and very small modules with a single value of $t_G$.

We need to be conservative when selecting the gene-score threshold because, if PISA misidentifies a module, elimination of its condition vector can lead to errors in other modules. Therefore, the number of genes included in modules due to noise should be very low. We have used a threshold of 7.0$\sigma$, which for a Gaussian distribution corresponds to about 3.9$\sigma$. The condition-score threshold $t_G$ must be adjusted to prevent false positives from the background: at low thresholds, a small module would be lost among false positives; while at high thresholds, it is mathematically impossible to find a large module. One can run ISA with many different threshold coefficients $t_G$ in order to find more modules than available at any single threshold, however this results in a large number of false positives.

Within PISA, we eliminate the need to use multiple gene-score thresholds by specifying the threshold relative to the background alone, which we estimate using the mean, $\langle x \rangle$, and the standard deviation, $\sigma$, of the gene scores within the shortest interval that contains at least 70% of all the gene scores. By excluding extreme gene scores in this way, we minimize the influence of the module of interest itself on the means and standard deviations of gene scores (Fig. 3). As a test, we used $\sigma$ in place of $\sigma_G$ in ISA and found it to be very large and very small modules with a single value of $t_G$.

We need to be conservative when selecting the gene-score threshold because, if PISA misidentifies a module, elimination of its condition vector can lead to errors in other modules. Therefore, the number of genes included in modules due to noise should be very low. We have used a threshold of 7.0$\sigma$, which for a Gaussian distribution corresponds to about 3.9$\sigma$. The condition-score threshold $t_G$ must be adjusted to prevent false positives from the background: at low thresholds, a small module would be lost among false positives; while at high thresholds, it is mathematically impossible to find a large module. One can run ISA with many different threshold coefficients $t_G$ in order to find more modules than available at any single threshold, however this results in a large number of false positives.

Within PISA, we eliminate the need to use multiple gene-score thresholds by specifying the threshold relative to the background alone, which we estimate using the mean, $\langle x \rangle$, and the standard deviation, $\sigma$, of the gene scores within the shortest interval that contains at least 70% of all the gene scores. By excluding extreme gene scores in this way, we minimize the influence of the module of interest itself on the means and standard deviations of gene scores (Fig. 3). As a test, we used $\sigma$ in place of $\sigma_G$ in ISA and found it to be very large and very small modules with a single value of $t_G$.

We need to be conservative when selecting the gene-score threshold because, if PISA misidentifies a module, elimination of its condition vector can lead to errors in other modules. Therefore, the number of genes included in modules due to noise should be very low. We have used a threshold of 7.0$\sigma$, which for a Gaussian distribution corresponds to about 3.9$\sigma$. The condition-score threshold $t_G$ must be adjusted to prevent false positives from the background: at low thresholds, a small module would be lost among false positives; while at high thresholds, it is mathematically impossible to find a large module. One can run ISA with many different threshold coefficients $t_G$ in order to find more modules than available at any single threshold, however this results in a large number of false positives.

Within PISA, we eliminate the need to use multiple gene-score thresholds by specifying the threshold relative to the background alone, which we estimate using the mean, $\langle x \rangle$, and the standard deviation, $\sigma$, of the gene scores within the shortest interval that contains at least 70% of all the gene scores. By excluding extreme gene scores in this way, we minimize the influence of the module of interest itself on the means and standard deviations of gene scores (Fig. 3). As a test, we used $\sigma$ in place of $\sigma_G$ in ISA and found it to be very large and very small modules with a single value of $t_G$.

We need to be conservative when selecting the gene-score threshold because, if PISA misidentifies a module, elimination of its condition vector can lead to errors in other modules. Therefore, the number of genes included in modules due to noise should be very low. We have used a threshold of 7.0$\sigma$, which for a Gaussian distribution corresponds to about 3.9$\sigma$. The condition-score threshold $t_G$ must be adjusted to prevent false positives from the background: at low thresholds, a small module would be lost among false positives; while at high thresholds, it is mathematically impossible to find a large module. One can run ISA with many different threshold coefficients $t_G$ in order to find more modules than available at any single threshold, however this results in a large number of false positives.
chance of including a gene due to noise is about $10^{-4}$ per gene, e.g. with the 6206 genes in the yeast data set, the average number of genes included by mistake in each module would be about 0.62. Using a high threshold means that we may miss genes that should belong to a module, however this is less risky than including genes by mistake. As PISA proceeds by eliminating condition-score vectors, it does not matter whether we identify all the genes in a module, as long as the condition-score vector is accurate. Potentially, once PISA has finished, one could easily see which genes would be included when using various gene-score thresholds for the same condition-score vector.

ISA only considers sets of genes that have high gene scores, i.e. positive signs. As discussed in Ihmels et al. (2002), this can lead to two modules that are regulated by the same conditions but with opposite sign. In contrast, PISA includes all genes with sufficiently extreme scores in a single module, and the relative signs of gene scores specify whether the genes are coregulated or counter-regulated.

2.3.3 Implementation of PISA To begin, PISA requires a matrix $E$ of log-ratio gene expression data, with zero average for each condition. Two matrices are obtained from $E$: the first $E_C$ is normalized for each gene

$$(\langle E_G \rangle_{C})_{k} = 0, (\langle E_G \rangle_{C})_{k} = 1 \quad \forall g \in G.$$  

Normalization of $E_C$ is essential so that the gene-score threshold can be applied to all genes on an equal footing. The second matrix $E$ is obtained from $E_C$ by normalizing for each condition. $$(\langle E_G \rangle_{C})_{k} = 1, \quad \text{where } E_G \text{ denotes the initial } E_C.$$  

(Note that this is essentially the opposite of the notation used in Bergmann et al., 2003.) PISA consists of a large number of steps (typically 10,000). In each step, we apply a modified version of ISA (PISAspect, see below), and if a module is found during the step, we remove from $E_C$ the components along the module’s condition-score vector $s^C$:  

$$E_C^{\text{new}} = E_C - E_C s^C (s^C)^T |s^C|^2.$$  

As PISA progresses, new modules are found less and less frequently. For example, one run of 10,000 steps found 779 preliminary (see below) modules, and 442 of them were found in the first 1000 steps. As the later modules are also generally smaller and less reliable, the exact number of steps is not very important.

2.3.4 PISAspect As input, a step of PISA requires the two matrices $E_C$ and $E_G$. We start each application of PISAspect by generating a random set of genes $G_0$ and a corresponding gene vector $m_0^G$:

$$(m_0^G)_i = \begin{cases} 1 & g \in G_0 \\ 0 & g \not\in G_0 \end{cases}.$$  

Each iteration $i$ within PISAspect consists of multiplying the transpose of $E_G$ by the gene vector $m_i^G$ to produce the condition-score vector $s^C_i$:

$$s^C_i = E_G^T m_i^G.$$  

and then multiplying $E_C$ by the normalized condition-score vector to produce the gene-score vector $s^G_i$:

$$s^G_i = E_C s^C_i |s^C_i|^2.$$  

From $s^G_i$, one calculates the gene vector $m_{i+1}^G$ for the next iteration:

$$(m_{i+1}^G)_i = \left( s^G_i ight)_i \Theta \left( \left| s^C_i \right| - \left( \left| s^C_i \right| \right)^{70\%}/\left| \left| s^C_i \right| \right|^{30\%} \right).$$  

We iterate until one of three conditions is met: (1) $(m_0^G)_i$ and $(m_0^G)_i$ have the same sign (0, + or −) for all $g$, (2) the iteration number is $i = 20$, or (3) fewer than two genes have non-zero weight. Criterion (1) indicates convergence to a fixed point, (2) handles limit cycles (see Section S.2), and (3) indicates failure to find a module. If fewer than five genes have non-zero weight, the result is discarded, otherwise we have found a module with condition-score vector $s^C = s^C_i$, gene-score vector $s^G_i = s^G_i$, and gene vector $m^G = m_{i+1}^G$. The module is then stored as a ‘preliminary module’ (see below), and $E_C$ is updated according to Equation (7).

We chose a threshold coefficient $c_i = 7.0$ so that the expected number of genes included in each module due to background noise would be less than 1. However, with this high threshold, starting from a random set of genes there was only a very low chance that two or more genes would score above the threshold in the first iteration. To increase the chance of finding a module, we used a different formula for $m^G_i$, i.e. for the first iteration only. Instead of selecting just those genes with scores above the threshold, we kept a random number $2 \leq n < 51$ of the genes with the most extreme scores. This procedure was generally adequate to produce a correlated set of genes for the next iteration.

PISAspect is very similar to an application of SVD to find an eigenarray/eigengene pair. The key difference is the gene threshold in PISA which requires the gene vector (eigenarray) to be sparse.

2.3.5 Consistent modules ISA typically finds many different fixed points corresponding to the same module, each differing by a few genes. PISA only finds each module once during a run, but the precise genes in the module depend on the random input set of genes and also on which modules were already found and eliminated. Furthermore, PISA sometimes finds a module by itself, while other times it may find the module joined with another module, or PISA may find only part of a module, or not find the module at all. To get a reliable set of modules, it was necessary to perform a number of runs of PISA and identify the modules that were consistent from run to run.

To identify consistent modules, we first tabulated preliminary modules—transcription modules found by individual runs of PISA. A preliminary module $P$ contributes to a consistent module $C$ if $P$ contains more than half the genes in $C$, regardless of gene-score sign, and these genes constitute at least 20% of the genes in $P$. $|P \cap C| > 0.5 |C| \land |P \cap C| > 0.2 |P|$ A gene is included in the consistent module if the gene occurs in more than 50% of the contributing preliminary modules, always with the same gene-score sign. We found the consistent modules by iteratively applying these criteria until we reached a fixed point, starting from all pairs of preliminary modules.

2.3.6 Correlations between condition-score vectors Once we identified a consistent module, $m^G$, we calculated the raw condition-score vector $r = E_G^T m^G$, using the initial value of the gene-expression data matrix $E_C$. From the $r$ we evaluated the condition correlations $r \cdot r'/|r||r'|$ between different modules. Additional details of the algorithm PISA are discussed in the supporting material.

2.4 p-Values

Given a set containing $m$ genes out of the total of $N_G$, the $p$-value for having at least $n$ genes in common with a Gene Ontology (GO) category containing $c$ of the $N_G$ genes is

$$p = \sum_{i=n}^{m} \binom{m}{i} \binom{N_G-i}{N_G-c}.$$  

This is not an issue in ISA, where the condition threshold helps to pick out the signal—which is possibly very small—from the noise.

7 The values 50, 20 and 50% used are subjective criteria for how consistent modules should be. However, the results are not very sensitive to these values.

8 While this approach may not be fully exhaustive, any consistent module missed by this approach is likely to be a variant of another consistent module or a marginal case.
We ignore any genes that are not present in our expression data when counting \( c \).

**3 RESULTS**

We applied PISA to the yeast data set used in Bergmann et al. (2003), which consists of log-ratio gene-expression data for \( N_G = 6206 \) genes and \( N_C = 987 \) experimental conditions (see Sections S.4 and S.5 for details). Normalization gives the matrices \( \mathbf{E}_G \) and \( \mathbf{E}_C \) (see Sections 2.3.3 and S.1 for details).

As a preliminary test, we repeatedly applied PISA to one fully scrambled version of the matrix \( \mathbf{E}_G \) (and the corresponding \( \mathbf{E}_C \)). From run to run, the algorithm identified many large modules derived almost entirely from a single condition, as expected in light of the broad distribution of the raw gene-expression data (Fig. S1). PISA also found many small modules, but these differed from one run to the next. We were able to eliminate both of these classes of false positives using filters for consistency, recurrence, and number of contributing conditions (Fig. S2; see Section S.3 for details).

We performed 30 runs of PISA on the yeast data set and identified the modules that appeared consistently, using the filters derived above. At the start of each run, only a few preliminary modules could be found with our single choice of gene threshold \( \theta_G \). Nevertheless, PISA did consistently find new preliminary modules after eliminating others, demonstrating that removing the condition vectors of found modules improves the signal to noise for the remaining ones. A total of 166 consistent modules passed the filters (PISA modules). Out of the 6206 genes included in the expression data, 2512 genes appeared in at least one PISA module, and more than 500 genes appeared in more than one PISA module.\(^{10}\) No genes appeared in more than four different PISA modules.

For most of the PISA modules, the genes were coregulated, i.e. all the gene scores had the same sign. (In contrast, the consistent modules that were eliminated by the filters often had about equal numbers of genes of either sign.) There were, however, a significant number of PISA modules with a few gene scores differing in sign from the rest, e.g. the arginine biosynthesis module described below. Furthermore, many of the PISA modules agreed closely with modules identified de novo at various thresholds, while other PISA modules were subsumed in the other modules. The inclusion of related processes, e.g. serine synthesis, in the module may be due to the ‘Borges effect’ (Mateos et al., 2002).

For many of the PISA modules, the genes were coregulated, i.e. all the gene scores had the same sign. (In contrast, the consistent modules that were eliminated by the filters often had about equal numbers of genes of either sign.) There were, however, a significant number of PISA modules with a few gene scores differing in sign from the rest, e.g. the arginine biosynthesis module described below. Furthermore, many of the PISA modules agreed closely with modules identified de novo at various thresholds, while other PISA modules were subsumed in the other modules. The inclusion of related processes, e.g. serine synthesis, in the module may be due to the ‘Borges effect’ (Mateos et al., 2002).

PISA found several small modules that agree very well with known gene regulation in yeast. For example, the arginine-biosynthesis module consists of ARG1, ARG3, ARG5,6, ARG8, CPA1, YOR302W, MEP3, CAR1 and CAR2; out of these CAR1 and CAR2 have negative gene scores, i.e. they are coregulated relative to the others. The first five genes are precisely the arginine-synthesis module consists of ARG1, ARG3, ARG5,6, ARG8, CPA1, YOR302W, MEP3, CAR1 and CAR2; out of these CAR1 and CAR2 have negative gene scores, i.e. they are coregulated relative to the others. The first five genes are precisely the arginine-synthesis module known to be repressed by arginine, while CAR1 and CAR2 are catabolic genes known to be induced by arginine (Messenguy and Dubois, 2000).

PISA also found a zinc (zap1-regulated) module even though the set of 987 conditions did not include zinc starvation. The set of genes in the module (ZRT1, ZRT2, ZRT3, ZAP1, YOL154, INO1, ADH4 and YNL254C) agree well with the highest-scoring genes in a separate microarray experiment comparing expression, under zinc starvation, of a ZAP1 mutant versus wild type (Lyons et al., 2000). For this module, the highest-scoring of the 987 conditions came from the Rosetta compendium (Hughes et al., 2000) of deletion mutants (see Fig. S9). Our identification of the unknown gene YNL254C as part of the zinc module, as well as the starvation experiments in Lyons et al. (2000) and direct transcription-factor-binding experiments (see below), all indicate that YNL254C is regulated by zap1, and probably functions in zinc starvation/uptake.

In order to evaluate the overall performance of PISA, we compared our PISA modules to the categories in the GO curated database (The Gene Ontology Consortium, 2001).\(^{11}\) For the set of genes in each of our modules we calculated the \( p \)-value for the overlap with the set of genes in every GO category (see Section 2). The \( p \)-value is the probability of obtaining the observed overlap by chance.

\(^{10}\) We have adjusted for the fact that some modules occur in several similar versions.

\(^{11}\) It is not clear to what extent the GO category definitions (molecular functions, cellular components and biological processes) correspond to the transcription modules we are searching for, which are characterized by coregulation. Thus, failure to find a good overlap with a GO category does not necessarily indicate that a module is not biologically relevant, but a very significant overlap does show biological relevance. Using GO \( p \)-values as a score should therefore be a reasonable way to compare the modules found using different approaches.
probability that an observed overlap occurred by chance. The lowest p-value we found was 5.7 \times 10^{-191}, for the GO category ‘cytosolic ribosome’, and we found p-values below 10^{-20} for more than 130 other GO categories. (The modules that were removed by our filters mostly did not have significant p-values.) Figure 5 shows a comparison between GO-category p-values for PISA and for ISA. While ISA does a somewhat better job at identifying large, strong modules, PISA does significantly better at finding small, weak modules. PISA also does better at producing accurate modules (we compared p-values in cases where at least 50% of the module genes belong to the GO category; data not shown). As shown in Figure S3, both algorithms perform much better than SVD.

We also used the p-values between our PISA modules and the GO categories to compare PISA to other means of identifying transcription modules. Specifically, we compared PISA to two different databases of genes predicted to be regulated by single transcription factors. Database ‘A’ contains genes that were enriched through immunoprecipitation with tagged transcriptional regulators (Lee et al., 2002), while database ‘B’ has genes sharing regulatory sequences derived by comparative genomics (Kellis et al., 2003). Figure 6 shows the p-values between GO and PISA compared to the p-values between GO and each of these two databases. The lower p-values for PISA indicate a consistently better agreement between GO and PISA than between GO and the other databases. While PISA may have a slight advantage in that it looks for overall coregulated genes as opposed to genes that share a single transcription factor, and this may be somewhat closer to the definitions of GO categories (biological processes, etc.), it is remarkable that there are no GO categories for which database A or B significantly outperforms PISA.

Compared to microarray data, database A and database B share one clear disadvantage: their binding sites are assigned to intergenic regions, and if the two genes bordering an intergenic region are divergently transcribed, then the databases do not identify which of the genes is regulated. In many cases, we found that by comparing sets of genes in database A to PISA modules, we could decide which of divergently transcribed genes were actually regulated. For example, database A lists six intergenic regions as binding site for zap1 at an internal p-value threshold of 10^{-5}, and four of these lie between divergently transcribed genes. However, five of the six intergenic regions border the genes ZRT1, ZRT2, ZRT3, ZAP1 and YNL254C, which PISA identifies as part of the zinc module.

Database A appears to have an additional source of false positives. Intergenic regions that are close to intergenic regions with very low p-values often have low p-values themselves, even when there is no apparent connection between the genes and no evidence of a binding site in the DNA sequence. For example, for the de novo purine-biosynthesis module, which is primarily regulated by the bas1 transcription factor, the intergenic region controlling GCV2 has the lowest p-value within database A, 1.1 \times 10^{-16}, and all the four closest intergenic regions have p-values below 10^{-5}. Comparison to PISA modules can help eliminate these potential false positives: out of the 29 genes assigned a p-value below 10^{-4} for bas1 binding in database A, 13 belong to a single PISA module, four others are divergently transcribed adjacent genes, and six others are genes transcribed from nearby intergenic regions.

4 DISCUSSION

PISA embodies a new approach to analysis of large gene-expression data sets. The central new feature in PISA is the robust elimination of transcription modules as they are found, by removing their condition-score vectors. Also new to PISA, compared to its precursors SA (Ihmels et al., 2002) and ISA (Bergmann et al., 2003), is the inclusion of both coregulated and counter-regulated genes in a single module, and the use of a single gene-score threshold.

Altogether, these new features result in an algorithm that can reliably identify both large and small regulatory modules, without supervision. We confirmed the performance of PISA by comparison to the GO database—PISA performed considerably better against GO than either high-throughput binding experiments or comparative

---

32We used the ISA modules included in the Matlab implementation available at http://barkai-serv.weizmann.ac.il/GroupPage/software.htm. This includes modules for threshold coefficients from 1.8 to 4.0.

33We used an internal p-value threshold of 0.001 for database A, as suggested in Lee et al. (2002).
Fig. 7. Correlations between modules identified by PISA (see text). The modules are ordered to form clusters; the full list is shown in Table S1 (same for both axes). This plot recapitulates many of the relationships shown in Ihmels et al. (2004), Figure 4: the three large, highly correlated areas shown above correspond to the three different trees of hierarchical clustering in that figure (lower left corner is amino acid synthesis, upper right corner is protein synthesis and mid-lower left is stress).

ACKNOWLEDGEMENTS

We wish to thank J.Ihmels and N.Barkai for sharing their dataset, and Rahul Kulkarni for valuable discussions. C.T. acknowledges support from the National Key Basic Research Project of China (No. 2003CB715900).

SUPPLEMENTARY DATA

Supplementary data for this paper are available at Bioinformatics online.

REFERENCES


