BIOINFORMATICS

Vol. 00 no. 00 2004
Pages 1–12

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

M. Kloster[1,2], C. Tang[2,3,*] and N. Wingreen[2,4]

[1]Department of Physics, Princeton University, Princeton, New Jersey 08544
[2]NEC Laboratories America, Inc., 4 Independence Way, Princeton, New Jersey 08540
[3]Center for Theoretical Biology, Peking University, Beijing 100871, China
[4]Department of Molecular Biology, Princeton University, Princeton, New Jersey 08544

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, S., Ihmels, J. and Barkai, N. (2002) *Phys. Rev. E* **67**, 031902], 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

Supporting material: Sections S.1–S.5, figures S1–S10 and table SI are available at ??

1 INTRODUCTION

The introduction of DNA microarray technology has made it possible to aquire 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 for a small, specific set of experimental conditions (Alon *et al.*, 1999; Eisen *et al.*, 1998; Tamayo *et al.*, 1999), these algorithms are less effective when applied to large data sets due to two well-recognized limitations. First, standard clustering algorithms assign each gene to a single cluster, while many genes in fact belong to multiple transcriptional regulons (Bittner *et al.*, 1999; Cheng and Church, 2000; Gasch and Eisen, 2002; Ihmels *et al.*, 2002). Second, each transcriptional regulon may only be active in a few experiments, and the remaining experiments will only contribute to the noise (Getz *et al.*, 2000; Cheng and Church, 2000; Ihmels it et al., 2002).

A number of approaches have been proposed to overcome one or both of these problems (Getz *et al.*, 2000; Califano *et al.*, 2000; Cheng and Church, 2000; Owen *et al.*, 2003; Gasch and Eisen, 2002; Lazzeroni and Owen, 2002). A particularly promising approach, the Signature Algorithm (SA) was introduced in Ihmels *et al.* (2002). Based on input sets of related genes, SA identifies "transcription modules" (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_{G} (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 modules found depend strongly on the value of a threshold coefficient t_{G} used in the algorithm. To find all the relevant modules, a large range of threshold values must be considered, and for each threshold the algorithm may find thousands of fixed points, many of which are spurious. While the largest, strongest modules are easily identified, among

*To whom correspondence should be addressed

© Oxford University Press 2004.

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 given by the activity of the various transcription factors in the cell[1]. If we assume that the effects of different transcription factors act multiplicatively on the expression level, then the relative expression levels of all the genes in an organism under a set of experiments ("conditions") are given by

$$\mathbf{E} = \sum_t \mathbf{g}_t \mathbf{c}_t^{\mathrm{T}} + \eta, \qquad (1)$$

where $\mathbf{E}_{gc}$ is the logarithmic expression ratio of gene g under condition c, the "gene vector" $\mathbf{g}_t$ specifies to what extent each gene is regulated by transcription factor t, and the "condition vector" $\mathbf{c}_t$ specifies the activity of that transcription factor in each condition (relative to its reference). η indicates noise. Together, we call corresponding gene and condition vectors a "transcription module" (TM), which may or may not actually correspond to a specific transcription factor.

The assumption of multiplicativity may be approximately true for lower organisms[2], but certainly does not capture the highly combinatorial regulation present in higher organisms. Nevertheless, Eq. (1) may be useful: each transcriptional module may then correspond to a relevant combination of transcription factors. There are also many other corrections, but Eq. (1) should still capture a large part of the gene-expression patterns of the organism.

Ideally, we would like to extract the full set of gene vectors and condition vectors based only on gene-expression data. The gene vectors describe the sets of genes that are coregulated at the most basic level in the organism, while the condition vectors describe how the organism responds to the different experimental conditions. Unfortunately, the decomposition of the matrix $\mathbf{E}$ given by Eq. (1) is not unique, even if there was no noise.

There are ways to find unique decompositions [as given by Eq. (1)] that have additional properties. One such approach is singular value decomposition (SVD; see *e.g.* Alter *et al.*, 2000), which leads to gene vectors (eigenarrays) that are all orthogonal to each other, as are the condition vectors (eigengenes). However, these properties do not match our biological expectations—different transcription factors may control substantially overlapping genes, and may also be active under many of the same experimental conditions. Also, 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. In particular, each transcription factor typically controls only a small subset of the genes in an organism, thus we expect the gene vectors to be sparse. A reasonable goal would thus be to find the simplest (*i.e.*, few TMs) decomposition for which the gene vectors are sufficiently sparse. A natural way to enforce sparse gene vectors is to introduce a threshold, such that no entry can be close to—but different from—zero.

While it is possible to search directly for a full decomposition with the desired properties, such an approach would be very computationally challenging. A more practical approach is to search for transcription modules one at a time, although correlations between different TMs makes also this a challenging problem. Ideally, in order to find the genes associated with a given transcription factor t in Eq. (1), we would want to look for a signal along a condition vector that has a large component along $\mathbf{c}_t$, but is orthogonal to the condition vectors of all other transcription modules, thus avoiding interfering signals. In practice, we can only ask that condition vectors are orthogonal to TMs we already know about.

2.2 The Algorithms SA/ISA

We briefly review the algorithms SA and ISA. A transcription module $\mathbf{M}$ can be specified by a condition vector ("experiment signature") $\mathbf{m}^{\mathrm{C}}$ and a gene vector ("gene signature") $\mathbf{m}^{\mathrm{G}}$, where nonzero entries in the vectors indicate conditions/genes that belong to the transcription module (TM).

[1] Post-transcriptional regulation by specific degradation of mRNA may also be considered to be a "transcription factor" effect in this context.

[2] Even for lower organisms, the ascription of one transcription module to each transcription factor is clearly not fully accurate. For instance, a transcription factor may repress some genes on the basis of its concentration only, while it may activate others depending on its phosphorylation level.

Given an appropriately normalized[3] matrix $\mathbf{E}$ of log-ratio gene-expression data and an input set G_{I} of genes, SA scores all the conditions in the data set according to how much each condition upregulates the genes in the input set (downregulation gives a negative score). The result is a condition-score vector $\mathbf{s}^{\mathrm{C}}$:

$$\mathbf{s}^{\mathrm{C}} \equiv \frac{\mathbf{E}^{\mathrm{T}}\mathbf{m}^{\mathrm{G}}_{\mathrm{in}}}{\left|\mathbf{m}^{\mathrm{G}}_{\mathrm{in}}\right|}, \tag{2}$$

where $\mathbf{E}^{\mathrm{T}}$ is the transpose of $\mathbf{E}$ and

$$(\mathbf{m}^{\mathrm{G}}_{\mathrm{in}})_g = \begin{cases} 1 & g \in G_I \\ 0 & g \notin G_I \end{cases} \tag{3}$$

is the gene vector corresponding to the input set. The entries of $\mathbf{s}^{\mathrm{C}}$ that are above/below a threshold $\pm t_{\mathrm{C}}$ constitute the condition vector $\mathbf{m}^{\mathrm{C}}$:

$$(\mathbf{m}^{\mathrm{C}})_c \equiv (\mathbf{s}^{\mathrm{C}})_c \cdot \Theta(\left|(\mathbf{s}^{\mathrm{C}})_c\right| - t_{\mathrm{C}}), \tag{4}$$

where $\Theta(x) = 1$ for $x \geq 0$ and $\Theta(x) = 0$ for $x < 0$.

Similarly, the gene-score vector $\mathbf{s}^{\mathrm{G}}$ measures how much each gene is upregulated by the conditions in $\mathbf{m}^{\mathrm{C}}$, using the entries of $\mathbf{m}^{\mathrm{C}}$ as weights:

$$\mathbf{s}^{\mathrm{G}} \equiv \frac{\mathbf{E}\,\mathbf{m}^{\mathrm{C}}}{|\mathbf{m}^{\mathrm{C}}|}. \tag{5}$$

The entries of the gene-score vector $\mathbf{s}^{\mathrm{G}}$ that are more than t_{G} standard deviations $\sigma_{\mathbf{s}^{\mathrm{G}}}$ above the mean gene score in the vector $\mathbf{s}^{\mathrm{G}}$ constitute the gene vector $\mathbf{m}^{\mathrm{G}}$:

$$(\mathbf{m}^{\mathrm{G}})_g \equiv (\mathbf{s}^{\mathrm{G}})_g \cdot \Theta((\mathbf{s}^{\mathrm{G}})_g - \langle(\mathbf{s}^{\mathrm{G}})_{g'}\rangle_{g'} - t_{\mathrm{G}}\sigma_{\mathbf{s}^{\mathrm{G}}}) \tag{6}$$

The Iterative Signature Algoritm (ISA) starts from a random set of genes and repeatedly applies SA, using $\mathbf{m}^{\mathrm{G}}$ as the input $\mathbf{m}^{\mathrm{G}}_{\mathrm{in}}$ for the next iteration, until a fixed point is reached. For the actual fixed point, the output $\mathbf{m}^{\mathrm{G}}$ would be exactly the same as the input $\mathbf{m}^{\mathrm{G}}_{\mathrm{in}}$ for an iteration of SA; in practice ISA stops when the same set of genes is selected in two consecutive iterations.

Both SA and ISA apply thresholds to both gene scores and condition scores. According to our discussion in section 2.1, this corresponds to an assumption that both gene vectors and condition vectors are 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 (t_{G}-dependent) limit on the fraction of genes that can be included in a module. The condition threshold, on the other hand, 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 less than 10% of the conditions, and some contain more than 80%.

[3] SA actually uses two matrices with different normalizations (Ihmels *et al.*, 2002).

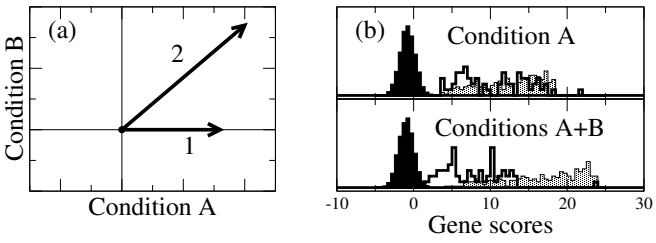

Fig. 1. A toy model with only two transcription modules (synthetic data). (a) Module 1 is upregulated under condition A, while module 2—a larger, stronger module—is upregulated under conditions A and B. The remaining (background) genes only show Gaussian noise. (b) Normalized histograms of the gene scores given by the Signature Algorithm (SA) for the background (solid fill), module 1 (solid line) and module 2 (dotted fill), when using the true condition vector for either module 1 (condition A) or module 2 (conditions A+B). Even starting with the true condition vectors, SA does not resolve the two modules. Nor can the Iterated Signature Algorithm (ISA) resolve module 1, even if it receives the module itself as input gene set, as the genes from module 2 have higher scores also for condition A (there is only one fixed point of ISA). Due to the noisy data, it is also impossible to separate the modules by varying the ISA gene threshold coefficient t_{G}.

As mentioned in section 2.1, different TMs are often correlated. This can contribute to a hierarchical clustering by ISA: For a low gene threshold coefficient t_{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, as shown in Fig. 1, it may be impossible for SA/ISA to resolve correlated modules regardless of which value is used for t_{G}.

2.3 The Algorithm PISA

2.3.1 Orthogonalization. Within PISA, each condition-score vector $\mathbf{s}^{\mathrm{C}}$ is required to be orthogonal to the condition-score vectors of all previously found transcription modules (TMs), as illustrated schematically in Fig. 2. Therefore, whenever PISA finds a TM and its associated condition-score vector $\mathbf{s}^{\mathrm{C}}$, the component along $\mathbf{s}^{\mathrm{C}}$ of each gene is removed from the gene expression matrix (see *Implementation of PISA* below). Returning to the example in Figs. 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 transcription modules *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

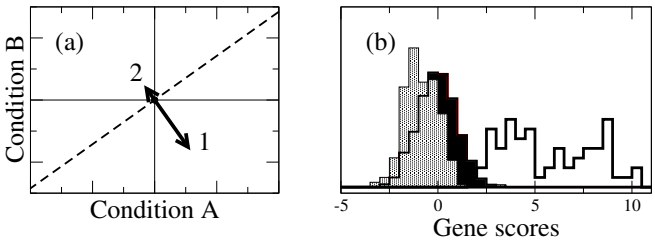

Fig. 2. Once the Progressive Itererative Signature Algorithm (PISA) has eliminated the combined module 1+2 from Fig. 1 (dashed line), the remaining signal makes it easy to separate the genes of module 1 from the genes of module 2. (a) Remaining signal for each module. (b) Actual gene scores for the new fixed point found by PISA. Genes of module 1 (solid line) have been separated from genes of module 2 (dotted fill) and the background (solid fill).

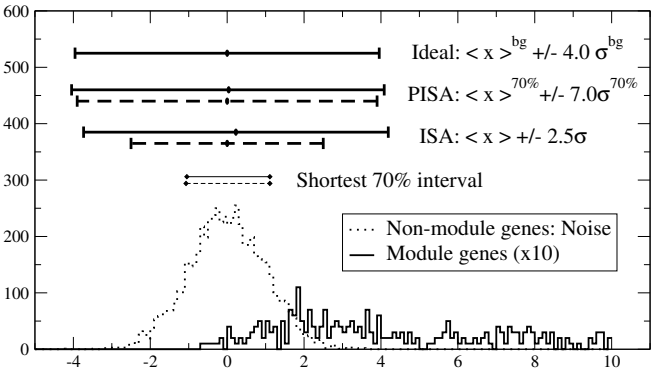

Fig. 3. Gene-score thresholds as used in ISA and in PISA algorithms (see text); for a synthetic gene-score distribution for 6206 genes, 300 of which belong to a module; calculated using all the genes (top, solid bars) or only the non-module genes (bottom, dashed bars). In ISA, the value $t_{\rm G} \approx 2.5$ that gives the desired (for PISA) threshold ± 4.0 in the presence of the module gives a much too low threshold if there is no strong module, while the threshold used in PISA is only weakly module-dependent. The non-module (background) genes' scores follow a normal distribution; $\langle x \rangle^{\rm bg} = 0,\ \sigma^{\rm bg} = 1$.

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 Eq. (1), but the resulting condition vectors are of course all orthogonal. A condition vector calculated from the initial value of the gene-expression-data matrix, as given in section 2.3.6 below, gives a much better description of the "real" transcription module.

2.3.2 The gene-score threshold. In ISA, the gene-score threshold is $t_G \sigma_{\mathbf{S}^{\rm G}}$, where the standard deviation $\sigma_{\mathbf{S}^{\rm G}}$ 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, $\sigma_{\mathbf{S}^{\rm G}}$ 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. This is not a significant issue in ISA, since one (independent of this) needs to run the algorithm with many different threshold coefficients $t_{\rm G}$ in order to find all modules. Within PISA, however, we wish to find all the modules using a single threshold—otherwise, without prior knowledge of the module one is searching for, it is difficult to know what $t_{\rm G}$ to use—which requires modifying the threshold definition.

We eliminate this problem in PISA by specifying the threshold relative to the background, which we estimate using the mean, $\langle x \rangle^{70\%}$, and the standard deviation, $\sigma^{70\%}$, 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 on the means and standard deviations of gene scores (Fig. 3). As a test, we used $\sigma^{70\%}$ in place of σ in ISA and found both very large and very small modules with a single value of t_G[4].

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 $7.0\sigma^{70\%}$, which for a Gaussian distribution corresponds to about 3.9σ. The 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

[4] It would still be necessary to use a large range of thresholds to find all the ISA modules; this is not just an artifact of the threshold definition.

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 $\mathbf{E}$ of log-ratio gene expression data, with zero average for each condition. Two matrices are obtained from $\mathbf{E}$: The first $\mathbf{E}_{\mathrm{G}}$ is normalized for each gene

$$\langle (\mathbf{E}_{\mathrm{G}})_{gc} \rangle_c = 0, \;\; \langle (\mathbf{E}_{\mathrm{G}})^2_{gc} \rangle_c = 1 \qquad \forall g \in G.$$

Normalization of $\mathbf{E}_{\mathrm{G}}$ is essential so that the gene-score threshold can be applied to all genes on an equal footing. The second matrix $\mathbf{E}_{\mathrm{C}}$ is obtained from $\mathbf{E}_{\mathrm{G}}$ by normalizing for each condition, $\langle (\mathbf{E}_{\mathrm{C},0})^2_{gc} \rangle_g = 1$, where $\mathbf{E}_{\mathrm{C},0}$ denotes the initial $\mathbf{E}_{\mathrm{C}}$. (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 (PISAstep, see below), and if it finds a module, we remove from $\mathbf{E}_{\mathrm{C}}$ the components along the module's condition-score vector $\mathbf{s}^{\mathrm{C}}$:

$$\mathbf{E}_{\mathrm{C}}^{\mathrm{new}} \equiv \mathbf{E}_{\mathrm{C}} - \mathbf{E}_{\mathrm{C}} \frac{\mathbf{s}^{\mathrm{C}} (\mathbf{s}^{\mathrm{C}})^{\mathrm{T}}}{|\mathbf{s}^{\mathrm{C}}|^2}. \qquad (7)$$

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 1,000 steps. As the later modules are also generally smaller and less reliable, the exact number of steps is not very important.

2.3.4 PISAstep. As input, a step of PISA requires the two matrices $\mathbf{E}_{\mathrm{C}}$ and $\mathbf{E}_{\mathrm{G}}$. We start each application of PISAstep by generating a random set of genes G_0 and a corresponding gene vector $\mathbf{m}_0^{\mathrm{G}}$:

$$(\mathbf{m}_0^{\mathrm{G}})_g = \begin{cases} 1 & g \in G_0 \\ 0 & g \notin G_0. \end{cases}$$

Each iteration i within PISAstep consists of multiplying the transpose of $\mathbf{E}_{\mathrm{C}}$ by the gene vector $\mathbf{m}_i^{\mathrm{G}}$ to produce the condition-score vector $\mathbf{s}_i^{\mathrm{C}}$:

$$\mathbf{s}_i^{\mathrm{C}} \equiv \mathbf{E}_{\mathrm{C}}^{\mathrm{T}} \mathbf{m}_i^{\mathrm{G}},$$

and then multiplying $\mathbf{E}_{\mathrm{G}}$ by the normalized condition-score vector to produce the gene-score vector $\mathbf{s}_i^{\mathrm{G}}$:

$$\mathbf{s}_i^{\mathrm{G}} \equiv \frac{\mathbf{E}_{\mathrm{G}} \mathbf{s}_i^{\mathrm{C}}}{|\mathbf{s}_i^{\mathrm{C}}|}.$$

From $\mathbf{s}_i^{\mathrm{G}}$, one calculates the gene vector $\mathbf{m}_{i+1}^{\mathrm{G}}$ for the next iteration:

$$(\mathbf{m}_{i+1}^{\mathrm{G}})_g \equiv (\mathbf{s}_i^{\mathrm{G}})_g \, \theta(|(\mathbf{s}_i^{\mathrm{G}})_g - \langle (\mathbf{s}_i^{\mathrm{G}})_{g'} \rangle_{g'}^{70\%}| - t_{\mathrm{G}} \sigma_{\mathbf{s}_i^{\mathrm{G}}}^{70\%}).$$

We iterate until: (a) $(\mathbf{m}_i^{\mathrm{G}})_g$ and $(\mathbf{m}_{i+1}^{\mathrm{G}})_g$ have the same sign (0, + or -) for all g, (b) the iteration number is $i = 20$, or (c) fewer than two genes have nonzero weight. Criterion (a) indicates convergence to a fixed point[5], (b) handles limit cycles (see section S.2), and (c) indicates failure to find a module. If fewer than five genes have nonzero weight, the result is discarded, otherwise we have found a module with condition-score vector $\mathbf{s}^{\mathrm{C}} = \mathbf{s}_i^{\mathrm{C}}$, gene-score vector $\mathbf{s}^{\mathrm{G}} = \mathbf{s}_i^{\mathrm{G}}$, and gene vector $\mathbf{m}^{\mathrm{G}} = \mathbf{m}_{i+1}^{\mathrm{G}}$. The module is then stored, and $\mathbf{E}_{\mathrm{C}}$ is updated according to Eq. (7).

We chose a threshold coefficient $t_{\mathrm{G}} = 7.0$ so that the expected number of genes included in each module due to background noise would be less than one. 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[6]. To increase the chance of finding a module, we used a different formula for $\mathbf{m}_1^{\mathrm{G}}$. Instead of selecting only genes with scores above the threshold, we kept a random number $2 \leq n \leq 51$ of the genes with the most extreme scores[7]. This procedure was generally adequate to produce a correlated set of genes for the next iteration.

PISAstep is very similar to one step of SVD; the key difference is the gene threshold in PISA.

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 $\mathbf{P}$ contributes to a consistent module $\mathbf{C}$ if $\mathbf{P}$ contains more than half the genes in $\mathbf{C}$, regardless of gene-score sign, and these genes constitute at least 20% of the genes in $\mathbf{P}$. ("$|\mathbf{P} \cap \mathbf{C}| > 0.5\,|\mathbf{C}| \;\; \wedge$

[5] If the gene set did not change, the distance to the fixed point is very small; further iteration generally only gives minimal changes.

[6] 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] 2 is the smallest number of genes that is interesting; the upper limit 51 is arbitrary as long as "large" sets are possible.

$|\mathbf{P} \cap \mathbf{C}| > 0.2\,|\mathbf{P}|$") 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[8]. We found the consistent modules by iteratively applying these criteria until we reached a fixed point, starting from all pairs of preliminary modules[9].

2.3.6 Correlations between condition-score vectors. Once we identified a consistent module, $\mathbf{m}^{\mathrm{G}}$, we calculated the raw condition-score vector $\mathbf{r} = \mathbf{E}_{\mathrm{C},0}^{\mathrm{T}}\mathbf{m}^{\mathrm{G}}$, using the initial value of the gene-expression-data matrix $\mathbf{E}_{\mathrm{C}}$. From the $\mathbf{r}$'s we evaluated the condition correlations $\mathbf{r} \cdot \mathbf{r}'/(|\mathbf{r}|\,|\mathbf{r}'|)$ between different modules.

Additional details 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

$$p = \sum_{i=n}^{\min\{c,m\}} \frac{\binom{c}{i}\binom{N_{\mathrm{G}}-c}{m-i}}{\binom{N_{\mathrm{G}}}{m}}, \qquad (8)$$

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 $N_{\mathrm{G}} = 6206$ genes and $N_{\mathrm{C}} = 987$ experimental conditions (see sections S.4; S.5 for details). Normalization gives the matrices $\mathbf{E}_{\mathrm{G}}$ and $\mathbf{E}_{\mathrm{C}}$ (see Methods; section S.1 for details).

As a preliminary test, we repeatedly applied PISA to one fully scrambled version of the matrix $\mathbf{E}_{\mathrm{G}}$ (and the corresponding $\mathbf{E}_{\mathrm{C}}$). 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 modules could be found with our single choice of gene threshold t_{G}. Nevertheless, PISA did consistently find new modules after eliminating others, demonstrating that removing the condition vectors of found modules improves the signal to noise for the remaining ones. 166 consistent modules passed the filters. Out of the 6206 genes included in the expression data, 2512 genes appeared in at least one module, and more than 500 genes appeared in more than one module[10]. No genes appeared in more than 4 different modules.

For most of the modules we found, the genes were coregulated, *i.e.* all the gene scores had the same sign. (In contrast, the modules that were eliminated by the filters often had about equal numbers of genes of either sign.) There were, however, a significant number of modules with a few gene scores differing in sign from the rest, *e.g.* the arginine biosynthesis module described below. Furthermore, many of the modules found by PISA agreed closely with modules identified by ISA at various thresholds, while other PISA modules were subsets of ISA modules. Some PISA modules, for example the de novo purine synthesis module (Fig. 4), were significantly more complete than the ones found by ISA (at any threshold).

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 counter-regulated relative to the others. The first five genes are precisely the arginine-synthesis genes 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 modules to the categories in the Gene Ontology (GO) curated database (The Gene Ontology Consortium, 2001)[11]. For the set of genes in each of our modules

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

[9] While this approach may not be fully exhaustive, any consistent module missed by this approach is almost certainly not of interest.

[10] We have adjusted for the fact that for some modules there are several versions that are very similar.

[11] It is not clear to what extent the GO category definitions (molecular functions, cellular components and biological processes) correspond to the transcriptional modules we are searching for, which are characterized by

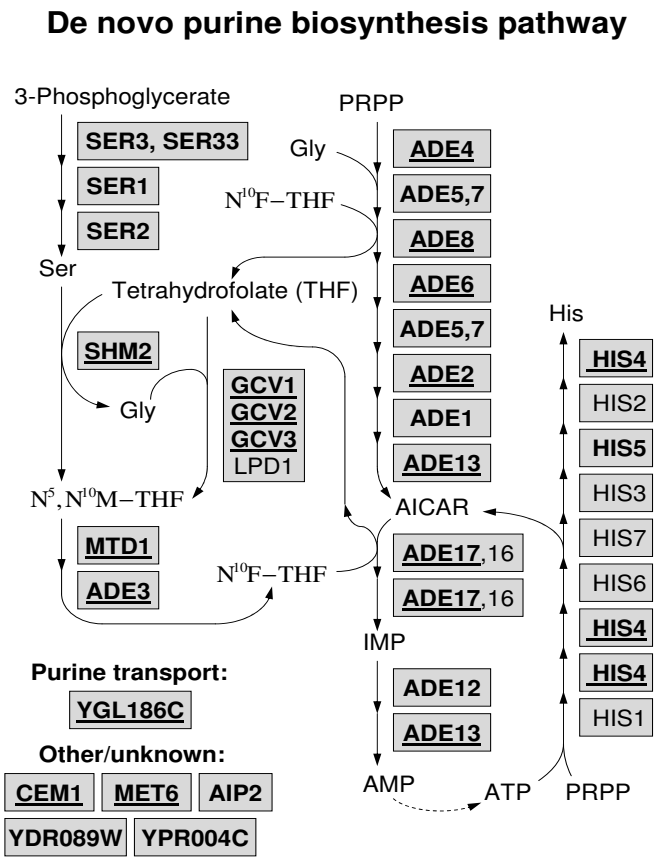

Fig. 4. The purine synthesis module found with PISA (genes shown in bold) contains all the key genes involved in de novo purine biosynthesis and associated one-carbon metabolism in yeast, as well as some of the genes involved in the closely connected histidine biosynthesis pathway. Purine synthesis is known to be regulated by the BAS1 transcription factor (Daignan-Fornier and Fink, 1992; Denis *et al.*, 1998); genes that are underlined have p-values below 0.001 for BAS1 binding in database A. Only key metabolites are shown. The inclusion of related processes, *e.g.* serine synthesis, in the module may be due to the "Borges effect" (Mateos *et al.*, 2002).

we calculated the p-value for the overlap with the set of genes in every GO category (see Methods). The p-value is the probability that an observed overlap occurred by chance. The lowest p-value we found was $5.7{\cdot}10^{-191}$, 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 such p-values

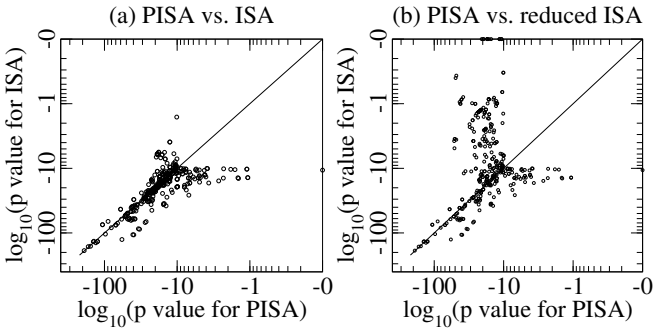

Fig. 5. Best p-values onto every Gene Ontology (GO) category with 500 or fewer genes. In each panel, we include only GO categories for which at least one p-value is below 10^{-10}. (a) 166 modules found by PISA vs. 778 modules found by ISA. ISA here does a slightly better job overall, but produces far more modules. (b) 166 PISA modules vs. the 215 most recurrect ISA modules from (a). PISA now does significantly better than ISA, with fewer modules. Furthermore, whenever ISA had a significantly lower p-value than PISA in (a), this is still the case in (b); thus the modules for which ISA does better are the highly recurrent large, strong modules, while PISA does a much better job at finding biologically relevant small, weak modules.

for PISA and for ISA[12]. 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 calculate p-values only when at least 50% of the module genes belong to the GO category; data not shown). As shown in Fig. 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.[13] 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

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 be a resonable way to compare different approaches as long as the module definitions used by the different approaches are about equally (dis)similar to the GO categories.

[12] We here use 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.

[13] We used an internal p-value threshold of 0.001 for Database A, as suggested in (Lee *et al.*, 2002).

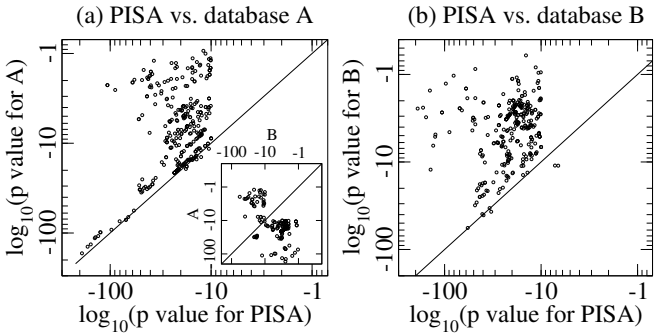

Fig. 6. Best p-values onto every Gene Ontology (GO) category with 500 or fewer genes. In each panel, we include only GO categories for which at least one p-value is below 10^{-10}. (a) PISA vs. Database A. (b) PISA vs. Database B. (a) inset: Database A vs. database B—there are very few GO categories onto which both A and B have low p-values.

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 6 intergenic regions as binding site for zap1 at an internal p-value threshold of 10^{-5}, and 4 of these lie between divergently transcribed genes. However, 5 of the 6 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, $1.1{\cdot}10^{-16}$, 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, 4 others are divergently transcribed adjacent genes, and 6 others are genes transcribed from nearby intergenic regions.

4 DISCUSSION

The Progressive Iterative Signature Algorithm (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 Gene Ontology (GO) database—PISA performed considerably better against GO than either high-throughput binding experiments or comparative genomics. PISA therefore provides a practical means to identify new regulatory modules and to add new genes to known modules.

Can PISA shed any light on the organization of gene expression beyond the level of individual transcription modules? In Bergmann *et al.* (2003), the authors argued that they could trace the relationship between modules from the effects of changing the threshold t_{G}, as done in greater detail in Ihmels *et al.* (2004). For instance, a large module might split into two smaller ones as t_{G} was increased. With PISA, we were able to use a more direct approach. Once we identified the modules, we computed the "raw" (*i.e.* pre-eliminations) condition-score vector $\mathbf{r}$ for each module, and from these raw condition-score vectors, we evaluated the condition correlations between modules (see Methods). Figure 7 shows the condition correlations between 40 of the modules that we can put a name to. A large, positive correlation between two modules can either indicate that the modules have many genes in common, *e.g.* the genes of the arginine-biosynthesis module are essentially a subset of the genes of the amino-acid-biosynthesis module, or, as in the toy model in Figs. 1 and 2, the modules have few/no genes in common, but the two sets of genes are similarly regulated under many conditions. In the toy model, the raw condition-score vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ correspond to the vectors in Fig. 1(a) and their correlation, $\mathbf{r_1} \cdot \mathbf{r_2}/(|\mathbf{r_1}|\,|\mathbf{r_2}|)$, is simply the cosine of the angle between them. A real example of this second type of correlation is provided by the ribosomal-protein module (107 genes) and the rRNA-processing module (80 genes). They have no genes in common, but the correlation between them is very high, 0.71.

To filter out false modules, we found it necessary to ignore all modules that depended only on a few conditions. As a result, true modules that were strongly regulated only in a few experiments could be missed. This suggests that experiments that affect many modules at once, in different patterns, are more useful than experiments that probe the effects of relatively simple perturbations. While the latter are easier to

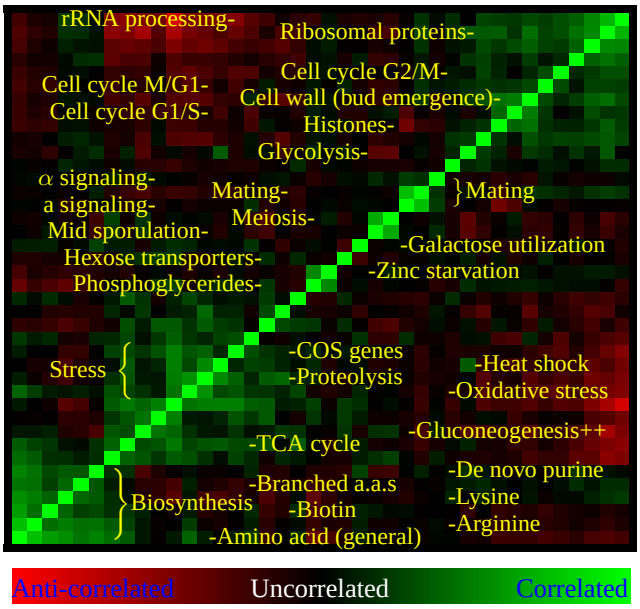

Fig. 7. Correlations between modules identified by PISA (see text). The modules are ordered to form clusters; the full list is shown in table SI (same for both axes). This plot recaptures many of the relationships shown in Ihmels *et al.* (2004), Fig. 4: The three large, highly correlated areas shown 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).

analyze one by one, there is more actual information in the former, and algorithms such as PISA can efficiently combine the results from many "complex" experiments to reveal the individual modules.

ACKNOWLEDGEMENTS

We wish to thank J. Ihmels and N. Barkai for sharing their data set, and Rahul Kulkarni for valuable discussions.

C.T. acknowledges support from the National Key Basic Research Project of China (No. 2003CB715900).

REFERENCES

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D. and Levine, A.J. (1999) Broad patterns of gene exporession revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci., USA*, **96**, 6745–6750.

Alter, O., Brown, P.O. and Botstein, D (2000) Singular value decomposition of genome-wide expression data processing and modeling. *Proc. Natl. Acad. Sci., USA*, **97**, 10101–10106.

Bergmann, S., Ihmels, J. and Barkai, N. (2003) Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys. Rev. E*, **67**, 031902.

Bittner, M., Meltzer, P. and Trent, J. (1999) Data analysis and integration: of steps and arrows. *Nature Genet.*, **22**, 213–215.

Califano, A., Stolovitzky, G. and Tu, Y. (2000) Analysis of gene expression microarrays for phenotype classification. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **8**, 75–85.

Cheng, Y. and Church, G. (2000) Biclustering of expression data. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **8**, 93–103.

Daignan-Fornier, B. and Fink, G.R. (1992) Coregulation of purine and histidine biosynthesis by the transcriptional activators BAS1 and BAS2. *Proc. Natl. Acad. Sci., USA* **89**, 6746–6750.

Denis, V., Boucherie, H., Monribot, C. and Daignan-Fornier, B. (1998) Role of the myb-like protein bas1p in Saccharomyces cerevisiae: a proteome analysis. *Mol. Microbiol.* **30**, 557–566.

Eisen, M.B., Spellman, P.T., Brown, P.O. and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci., USA*, **95**, 14863–14868.

Gasch, A. and Eisen, M.B. (2002) Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol.*, **3(11)**:research0059.1–0059.22.

Getz, G., Levine, E. and Domany, E. (2000) Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci., USA*, **97**, 12079–12084.

Hughes, T. R. *et al.* (2000) Functional discovery via a compendium of expression profiles. *Cell*, **102**, 109–126.

Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y. and Barkai, N. (2002) Revealing modular organization in the yeast transcriptional network. *Nature Genet.*, **31**, 370–377.

Ihmels, J., Ronen, L. and Barkai, N. (2004) Principles of transciptional control in the metabolic network of *Saccharomyces cerevisiae*. *Nature Biotech.*, **22**, 86–92.

Kellis, M., Patterson, N., Endrizzi, M., Birren, B. and Lander, E. S. (2003) Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, **423**, 241–254.

Lazzeroni, L. and Owen, A. (2002) Plain models for gene expression data. *Statistica Sinica*, **12**, 61–86.

Lee, T. I. *et al.* (2002) Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, **298**, 799–804.

Lyons, T. J., Gasch, A. P., Gaither, L. A., Botstein, D., Brown, P. O. and Eide, D. J. (2000), Genome-wide characterization of the Zap1p zinc-responsive regulon in yeast. *Proc. Natl. Acad. Sci., USA*, **97**, 7957–7962.

Mateos, A., Dopazo, J., Jansen, R., Tu, Y., Gerstein, M. and Stolovitzky, G. (2002), Systematic learning of gene functional classes from DNA array expression data by using multilayer perceptrons. *Genome Res.*, **12**, 1703–1715.

Messenguy, F. & Dubois, E. (2000) Regulation of arginine metabolism in *Saccharomyces cerevisiae*: a network of specific and pleiotropic proteins in response to multiple environmental signals. *Food tech. biotech.*, **38**, 277–285.

Owen, A.B., Stuart, J., Mach, K., Villeneuve, A.M. and Kim, S. (2003) A gene recommender algorithm to identify coexpressed genes in *C. elegans*. *Genome Res.*, **13**, 1828–1837.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E.S. and Golub, T.R. (1999) Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci., USA*, **96**, 2907–2912.

The Gene Ontology Consortium (2001) Creating the Gene Ontology resource: design and implementation. *Genome Res.*, **11**, 1425–1433.

SUPPORTING MATERIAL

S.1 Normalization

Here we review in detail the normalization procedure employed in PISA. The most obvious requirement for the normalization is that scores for different genes must be comparable. The procedure itself is as follows: Given a matrix $\mathbf{E}$ of log-ratio gene-expression data, we first set the average to zero for each condition,

$$(\mathbf{E}')_{gc} = (\mathbf{E})_{gc} - \langle(\mathbf{E})_{g'c}\rangle_{g'}, \tag{S1}$$

and then normalize to zero mean and unit variance for each gene, giving $\mathbf{E}_{\mathrm{G}}$, which is used in PISA to calculate gene scores:

$$(\mathbf{E}'')_{gc} = (\mathbf{E}')_{gc} - \langle(\mathbf{E}')_{gc'}\rangle_{c'} \tag{S2}$$

$$(\mathbf{E}_{\mathrm{G}})_{gc} = (\mathbf{E}'')_{gc}/\sqrt{\langle(\mathbf{E}'')^2_{gc'}\rangle_{c'}}. \tag{S3}$$

For this normalization to be consistent through the iterations in PISAstep, the different condition scores must also be comparable. To get the initial value $\mathbf{E}_{\mathrm{C},0}$ of the matrix used to calculate condition scores, we divide $\mathbf{E}_{\mathrm{G}}$ by the rms value for each condition:

$$(\mathbf{E}_{\mathrm{C},0})_{gc} = (\mathbf{E}_{\mathrm{G}})_{gc}/\sqrt{\langle(\mathbf{E}_{\mathrm{G}})^2_{g'c}\rangle_{g'}}. \tag{S4}$$

Note that a simple approach would be to normalize for both genes and conditions simultaneously and thus use only a single set of data[14]—this could be easily accomplished by alternately normalizing over conditions and genes a few times; the data converge quickly. There is, however, a risk of losing significant features of the data through excessive normalization. For some conditions, the typical change in expression levels may be very large, while for others it may be negligible, and it would be misleading to always normalize these to the same level; at the very least, this would give a lower signal to noise ratio. Therefore, we have chosen to normalize $\mathbf{E}_{\mathrm{G}}$ over genes but not conditions, allowing conditions with large changes in expression level to make a proportionately larger contribution to gene scores. For genes, however, it is reasonable to always normalize to the same level. If two genes are in the same module, then there is little reason to consider the gene with the larger dynamical range to be more reliable than the other. That is why we use $\mathbf{E}_{\mathrm{G}}$ to calculate $\mathbf{E}_{\mathrm{C},0}$.

Also note the difference between genes and conditions: The variance for a gene often depends on a small number of outlying values, and normalizing over genes prevents these from dominating. In contrast, the variance for a condition typically depends on many genes, and as such is a far more reliable quantity.

[14] If $\mathbf{E}_{\mathrm{G}} = \mathbf{E}_{\mathrm{C},0}$ initially, then it is equivalent to keep $\mathbf{E}_{\mathrm{G}}$ constant or use $\mathbf{E}_{\mathrm{G}} = \mathbf{E}_{\mathrm{C}}$, which is updated every time PISA finds a module.

S.2 Avoiding Positive Feedback

The basic principle of SA, or an iteration of ISA/PISAstep, is to find the set of genes whose expression profiles most resemble those of the genes in the input set, either for all conditions (PISAstep) or for a selected subset of conditions (SA/ISA). Of course, the gene whose expression profile most resembles that of a given gene is the gene itself, thus there is a potential for significant positive feedback. Adding one gene to the input set would typically increase the score of that gene far more than the score of any other gene. As a consequence of positive feedback, adding one gene to the gene vector of a fixed point would have a considerable chance of yielding another fixed point, and a small set of genes could be a fixed point even if the genes were completely uncorrelated.

In PISA, we only find each module (or combination) once for each run, and it is important to be as certain as possible that we have the correct genes. We avoid positive feedback by using leave-one-out scoring for genes that had nonzero weight at the start of the iteration, *i.e.* we remove the contribution from gene g from the condition scores $\mathbf{s}^{\mathrm{C}}_i$ before we use these scores to calculate the new score for gene g:

$$(\mathbf{s}^{\mathrm{G}}_i)_g \equiv \frac{(\mathbf{E}_{\mathrm{G}})_{g\_}[\mathbf{s}^{\mathrm{C}}_i - (\mathbf{E}^{\mathrm{T}}_{\mathrm{C}})_{\_g}(\mathbf{m}^{\mathrm{G}}_i)_g]}{\left|\mathbf{s}^{\mathrm{C}}_i - (\mathbf{E}^{\mathrm{T}}_{\mathrm{C}})_{\_g}(\mathbf{m}^{\mathrm{G}}_i)_g\right|},$$

where $(\mathbf{A})_{j\_}$ is row j of matrix $\mathbf{A}$, and $(\mathbf{A})_{\_j}$ is column j of matrix $\mathbf{A}$. With a Gaussian distribution of the background noise, this approach is very close to neutral, *i.e.* adding a gene will neither affect that gene's score, nor will it significantly change $\sigma^{70\%}$ of the gene-score distribution.

Without positive feedback, fixed points may be marginally stable or even unstable, *i.e.* a limit cycle, thus we do not require a true fixed point; we accept any gene vector reached after 20 iterations in PISAstep, as long as it contains at least 5 genes—empirically, if PISAstep has not converged in 20 iterations, then it has entered a limit cycle. An alternate criterion would be to check each iteration whether PISAstep has entered a limit cycle. An advantage of the fixed-iteration criterion is that this does not favor the state at which PISAstep would usually enter the limit cycle (which we see no reason to): whenever there is no clear reason to choose one result over another, we wish to sample all possibilities, which gives the post-processing algorithm better data to judge whether or not a module is reliable.

In SA/ISA, the authors do not eliminate positive feedback. Indeed it would be difficult to do so, as adding/removing a gene can change which conditions have scores exceeding the condition threshold. Apart from this complication, the feedback in SA/ISA is proportional to the number of conditions that make the threshold. For small modules, typically only a small fraction of the conditions have scores above the threshold, thus the feedback is lower than it would have been for PISA, which includes all conditions. For large modules, the

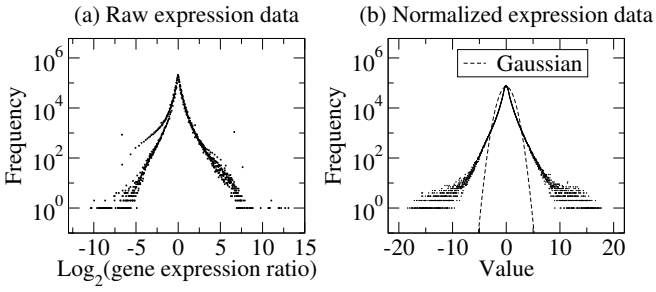

Fig. S1. Distributions of the yeast microarray data used (6206 genes/ORFs, 987 conditions). Roughly 10% of the data was invalid/missing (not included in the distributions). The distribution is sharply cusped and has long tails, both before and after normalization (Eqs. S1–S3).

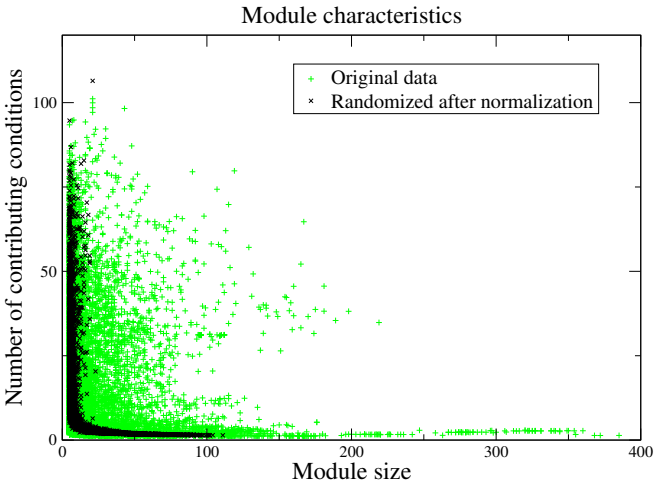

Fig. S2. The number of genes $n_{\mathbf{M}}^{\mathrm{G}}$ in a module $\mathbf{M}$ and the number of contributing conditions $n_{\mathbf{M}}^{\mathrm{C}}$ (see text) were two of the properties we used in our filters to eliminate false modules. PISA applied to a scrambled expression matrix (black) only yielded modules close to the axes (small $n_{\mathbf{M}}^{\mathrm{G}}$ or small $n_{\mathbf{M}}^{\mathrm{C}}$), while PISA run on the real data (green) yielded modules with both large $n_{\mathbf{M}}^{\mathrm{G}}$ and large $n_{\mathbf{M}}^{\mathrm{C}}$.

feedback is only a minor effect in the first place. Nevertheless, the total number of fixed points for ISA is huge due to positive feedback—at a gene threshold coefficient $t_{\mathrm{G}} = 4.0$, there are, at a minimum, more than a million fixed points.

S.3 Filters

We chose the gene-score threshold as $7.0\sigma^{70\%}$ so that, on average, less than one gene would be included in a module purely due to background noise. This estimate assumed that the background noise had a Gaussian distribution. For most modules, the gene scores are the sums of contributions from many different conditions, and if these contributions are independent, as they should be for background noise, then the total background noise will have approximately a Gaussian distribution, regardless of the distribution for a single condition (central limit theorem). For modules that derive almost entirely from one or very few conditions, however, the distribution of gene scores may not be Gaussian.

While we do not know the true distribution of the background noise, it is reasonable to use the full distribution of the data as a worst case scenario. As shown in Fig. S1, this distribution is far from Gaussian: it has a fairly sharp cusp at zero and long tails, even after normalization. For this distribution, more than 3% of the values are outside the threshold $\pm 7.0\sigma^{70\%}$ (this is partially because the long tails contain many genes, and partially because $\sigma^{70\%}$ is small due to the sharp cusp), *i.e.* with a gene-expression matrix randomly drawn from this distribution, for any single condition one would expect to find a module with about 200 genes!

We applied PISA to a matrix $\mathbf{E}_{\mathrm{G}}$ that had been fully scrambled after normalization[15]. As shown in Fig. S2, PISA found many large modules that were based almost entirely on a single condition (however, as the modules were not based on *only* one condition, they were not as large as our estimate of 200, above), whereas modules based on many conditions were much smaller. We also applied PISA to a random matrix generated from a Gaussian distribution, and in that case PISA did not find any large modules (in 30 runs, PISA found 8 modules with 20 or more genes; the largest contained 26 genes). In both cases, the small modules found by PISA varied from run to run.

In order to eliminate these false modules we introduced a set of filters. For each preliminary module $\mathbf{M}$ we calculate the "number of contributing conditions", given as $n_{\mathbf{M}}^{\mathrm{C}} = \sum_c (\mathbf{s}^{\mathrm{C}})_c^2 / (\max\{(\mathbf{s}^{\mathrm{C}})_c\})^2$. We ignored any module for which the median of the numbers of contributing conditions for its preliminary modules was below 4, 5 or 7.5, depending on the size of the module[16] (these thresholds worked well; they are somewhat above the threshold required to remove the false positives for the scrambled matrix). A second filter was based on the "consistency", defined as the fraction of the genes in a preliminary modules that are in the full module times the fraction of the genes in the full module that are in the preliminary module. We ignored any module with average consistency below 0.3, as well as modules with average consistency below 0.5 that had less than 20

[15] Scrambling the matrix *after* normalization ensured that the distribution remained the same. The data were no longer exactly normalized for each gene, but the deviations were insignificant. Scrambling the data before normalization gave similar results.

[16] As shown in Fig. S2, $n_{\mathbf{M}}^{\mathrm{C}}$ tends to be smaller for larger modules (with fewer genes, it is more difficult to "specify" a single condition), thus we ignored modules with 40 or more genes if $n_{\mathbf{M}}^{\mathrm{C}} < 5$ and modules with 10-39 genes if $n_{\mathbf{M}}^{\mathrm{C}} < 7.5$. For modules with less than 10 genes, $n_{\mathbf{M}}^{\mathrm{C}}$ is no longer a very good indicator of whether or not the module is reliable, so we only ignored these modules if $n_{\mathbf{M}}^{\mathrm{C}} < 4$; for these small modules, the consistency requirements are much more important.

contributing preliminary modules. We also ignored all modules that had fewer than 5 genes or fewer than 5 contributing preliminary modules. These filters removed all but 14 of the 643 modules found by PISA when applied to the scrambled matrix.

The values used above in the filters are partially based on the distributions for randomized data (as shown in Fig. S2 for two descriptors), but have been manually adjusted to better separate interesting modules from apparent false positives in the real data. Increasing the threshold values will initially eliminate only a few interesting modules, while lowering them will admit a large number of apparent false positives. (Modules that have no obvious biological relevance and that have about equal numbers of genes with either sign are here assumed to be false positives, while "interesting" modules are ones for which the genes appear to be functionally related.) While the current set of filters do a good job of separating interesting modules from false positives, they are probably far from optimal.

S.4 Missing values

About 10% of the 6206x987 data values are missing, and these are not all randomly distributed among the genes/conditions. In order to avoid biases for or against including a gene in a module based on how many missing values that gene has, we exclude the conditions for which a given gene has missing values when calculating that gene's score

$$\mathbf{s}_i^{\mathrm{G}} \equiv \frac{\mathbf{E}_{\mathrm{G}}\mathbf{s}_i^{\mathrm{C}}}{\left|\mathbf{s}_i^{\mathrm{C}}\right|},$$

both when multiplying with $\mathbf{E}_{\mathrm{G}}$ and when calculating $\left|\mathbf{s}_i^{\mathrm{C}}\right|$. In $\mathbf{E}_{\mathrm{C},0}$, we set missing values equal to zero. (For the orthogonalization process to work properly, all values must be defined.)

S.5 Data set

The yeast data set we use is based on the data set used in Bergmann *et al.* (2003), which contained 6206 genes and 1011 experimental entries (essentially all available yeast microarray data at the time). However, 20 of these entries did not contain original experimental data, and an additional 4 entries contained data in a format that could not reliably be converted to $\log_2$ ratios. Another 14 entries contained data in a wrong format, but we were able to convert this data to $\log_2$ ratios, giving a total of 987 valid conditions.

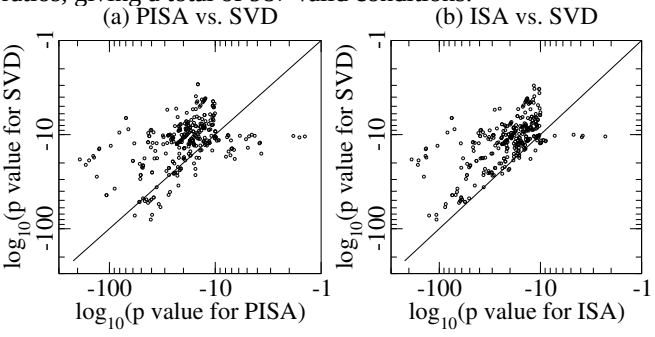

Fig. S3. Best p-values onto every Gene Ontology (GO) category with 500 or fewer genes. In each panel, we include only GO categories for which at least one p-value is below 10^{-10}. SVD modules are found by taking the 5, 10, 15, ..., 200 genes with the highest (lowest) entries in the eigenarrays, for a total of $987 \cdot 80 = 78,960$ modules. (a) 166 modules found by PISA vs. SVD. (b) 778 ISA modules vs. SVD. PISA and ISA are both clearly superior to SVD; for ISA there are hardly any GO categories for which SVD does better, even though SVD here has a hundred times as many modules as ISA. The eigenarrays used here are the eigenvectors of $\mathbf{E}_{\mathrm{G}}\mathbf{E}_{\mathrm{C},0}^{\mathrm{T}}$—SVD then corresponds to PISA without a gene threshold.

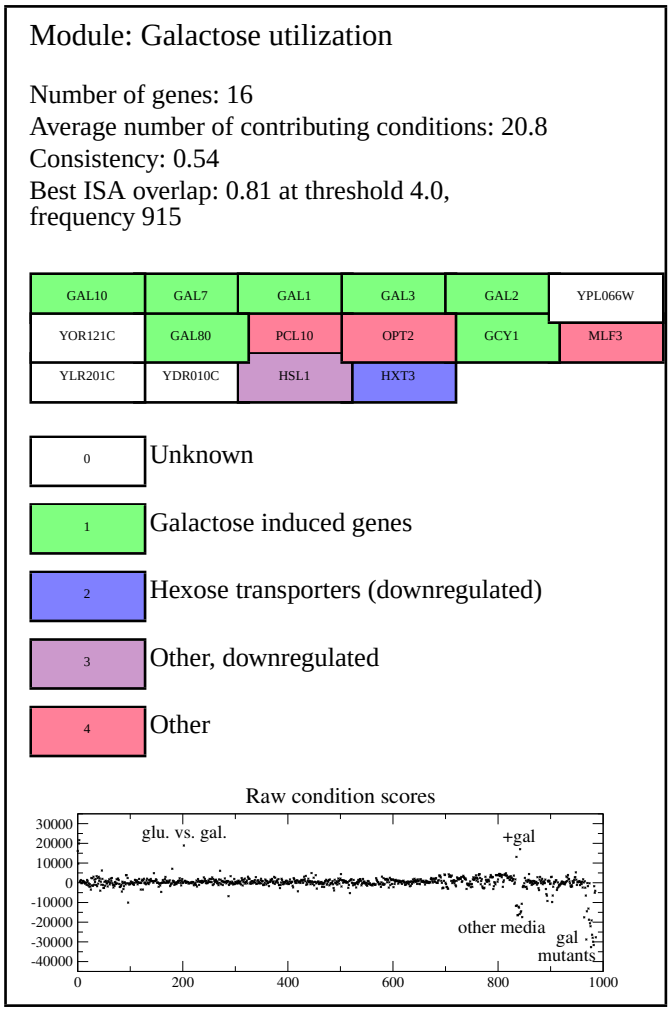

Fig. S4. The galactose induced module found with PISA. This module turns on GAL genes and also, as a weaker effect, represses a number of hexose transporters.

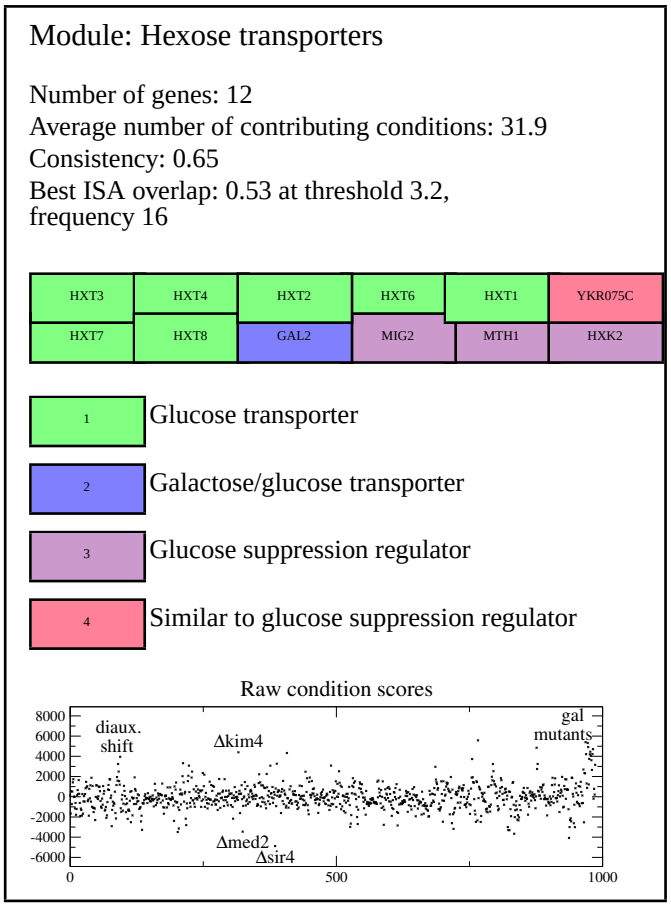

Fig. S5. The hexose transporter module found with PISA. In this module (which is consistently found after the galactose induced module), the hexose transporter genes are co-regulated with GAL2, the galactose permease, whereas they were counter-regulated in the galactose induced module.

Module: De novo purine biosynthesis

Number of genes: 27
Average number of contributing conditions: 17.4
Consistency: 0.63
Best ISA overlap: 0.67 at threshold 5.0,
frequency 12

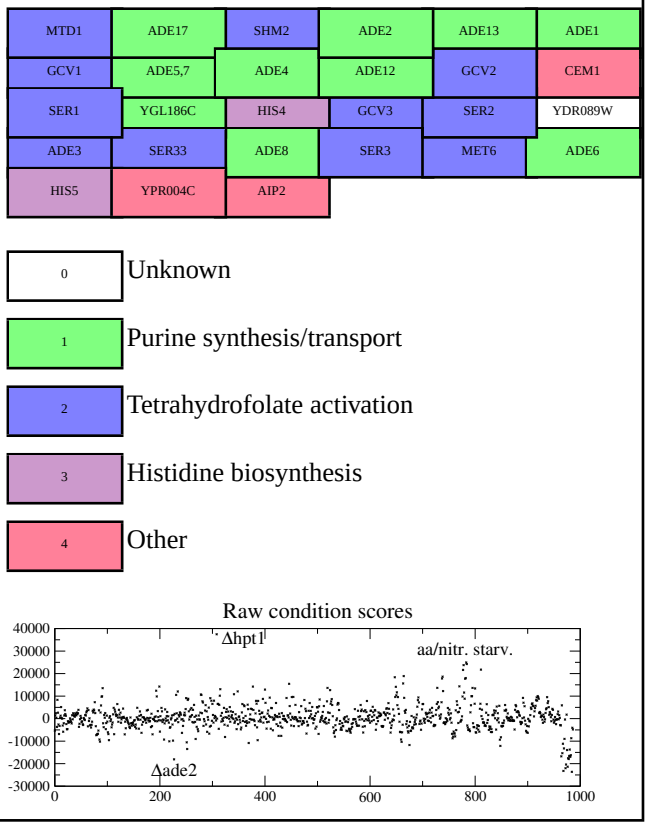

Fig. S6. The de novo purine synthesis module found with PISA.

Module: Peroxide shock

Number of genes: 62
Average number of contributing conditions: 25.7
Consistency: 0.78
Best ISA overlap: 0.35 at threshold 3.4,
frequency 621

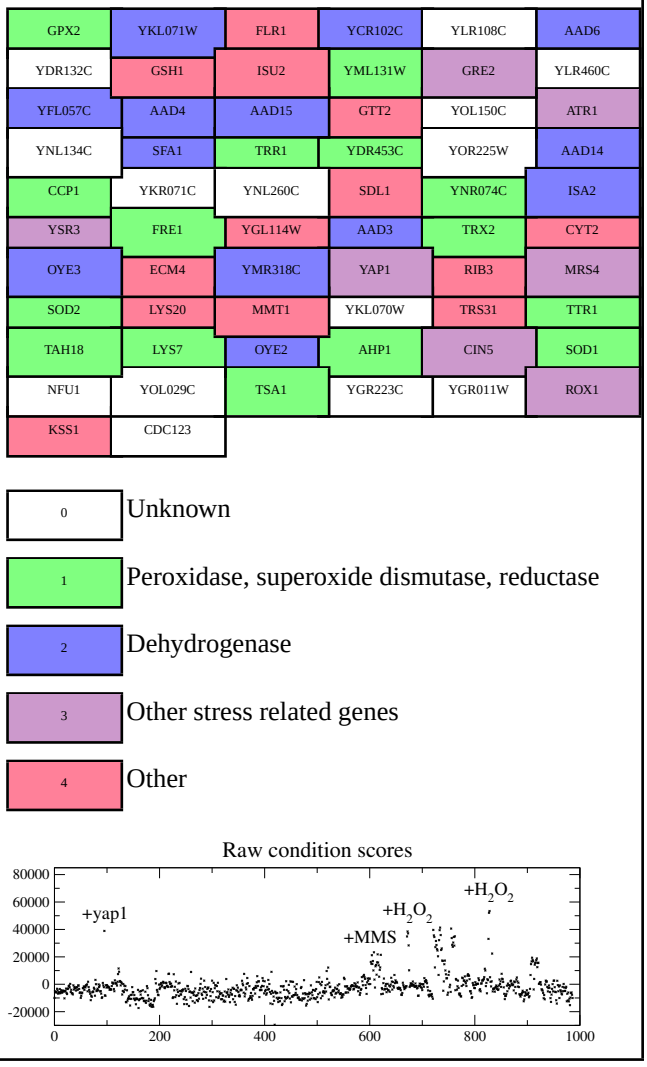

Fig. S7. The oxidative stress response module found with PISA.

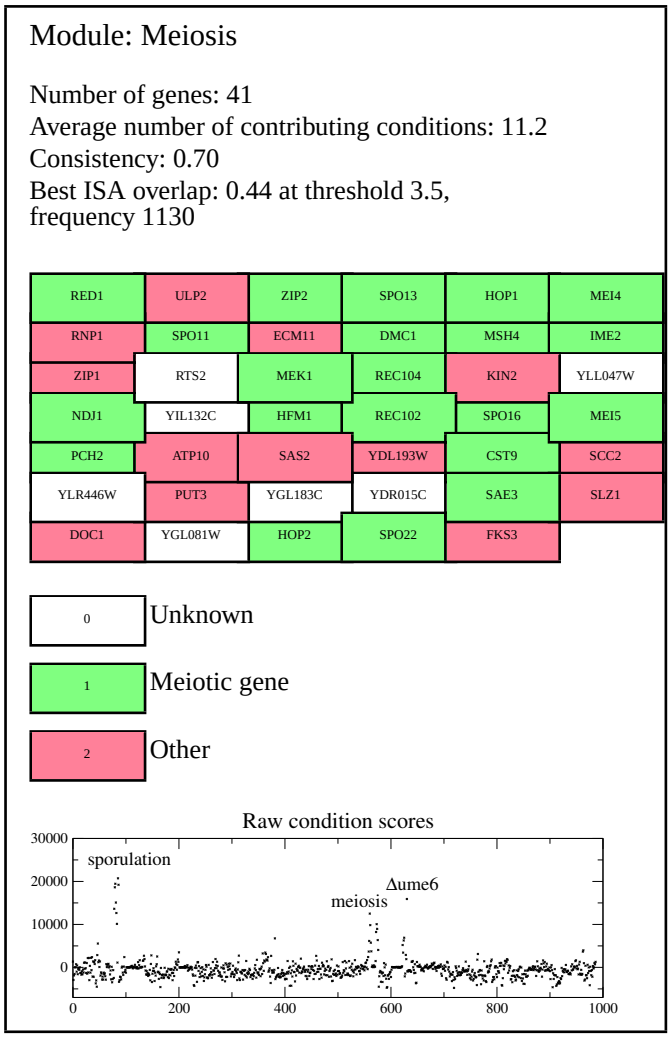

Fig. S8. The meiosis module found with PISA. This module is significantly more complete than the modules of comparable size found by ISA.

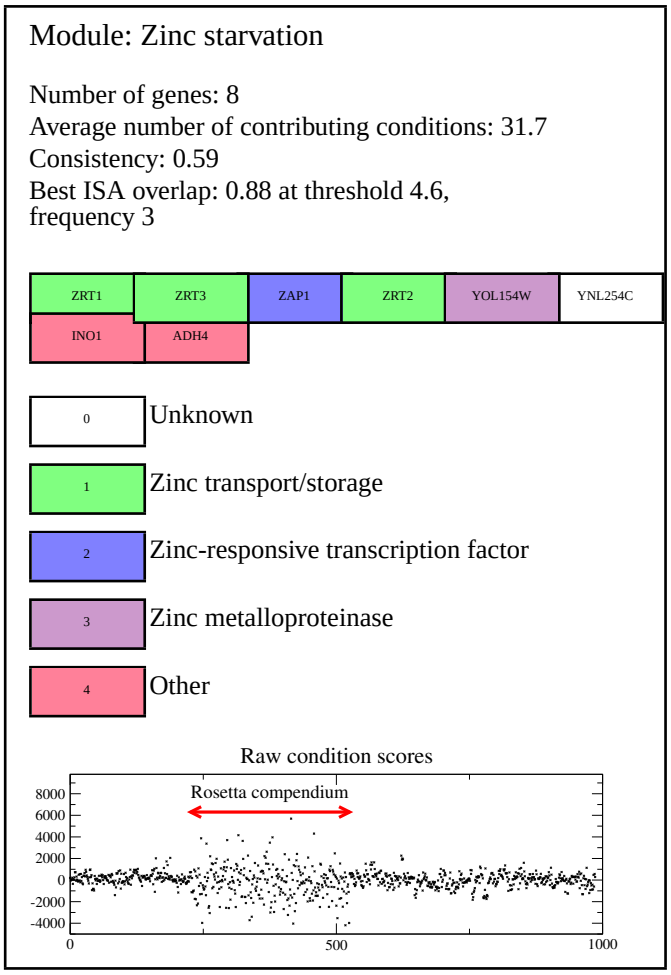

Fig. S9. The zinc module found with PISA. This module has a high overlap with the group of genes bound by ZAP1 in database A (at p-value 0.001): The ZRT1, ZRT2, ZRT3, ZAP1 and YNL254C genes make up 5 of the 6 lowest p-values (counting each pair of divergently transcribed genes only once), and the remaining hits from database A (most with p-values above 10^{-4}) are likely to be mostly false positives. Based on this, it seems very likely than YNL254C, if functional, is regulated by and related to zinc. (ADH4 has also been shown to be zinc-regulated elsewhere.)

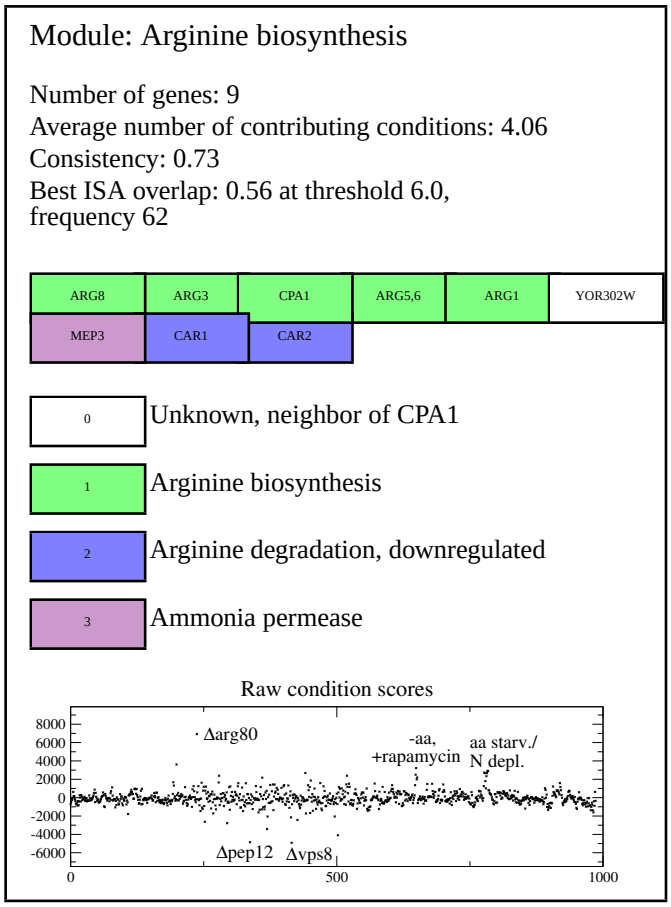

Fig. S10. The arginine regulated module found with PISA. The module agrees very well with what is known about regulation of arginine metabolism [F. Mesenguy and E. Dubois (2000) *Food tech. bio.* **38**, 277-285]: ARG1, ARG3, ARG5,6 and ARG8 are repressed by arginine through the Arg80/Arg81/Mcm1 complex, while CAR1 and CAR2 are activated by the same complex. We also find CPA1, which is claimed to be regulated by arginine at the translational level—the mRNA is destabilized by a small peptide in the presence of arginine. However, database A indicates that ARG1, ARG3, ARG5,6, ARG8 and CPA1 are all bound by the Arg80/Arg81/Mcm1 complex.

Function	# genes	# cond.	Cons.	Overlap w/ISA	Best t_G	Freq.
Amino acid biosynthesis	108	32.2	0.94	0.85	3.4	20212
Arginine biosynthesis	9	4.0	0.73	0.56	6.0	62
Biotin synthesis & transport	6	6.8	0.79	0.67	5.5	7
Lysine biosynthesis	11	9.3	0.71	0.82	4.6	10
Branched amino acid biosynthesis	29	10.7	0.74	0.47	3.2	70
De novo purine biosynthesis	27	16.6	0.63	0.67	5.0	12
Sulfur/nitrogen metabolism	47	15.8	0.59	0.62	2.6	1205
Citric acid cycle	12	14.4	0.54	0.67	3.0	20
Gluconeogenesis, fatty acid beta-oxidation	38	17.7	0.73	0.61	2.9	264
Oxidative phosphorylation	46	39.4	0.78	0.93	3.4	1666
Trehalose & hexose metabolism/conversion	23	51.5	0.60	0.61	3.1	524
Oxidative stress response	62	25.7	0.78	0.35	3.4	621
Proteolysis	23	85.1	0.69	0.88	3.9	352
Heat shock	54	48.3	0.68	0.44	3.2	12
COS genes	11	12.4	0.58	1.00	3.3	756
Calcium-calmodulin related	37	32.3	0.63	0.81	3.5	610
Mitochondrial ribosomal genes	78	42.4	0.64	0.83	3.0	3941
Transcription (RNA polymerase etc.)++	34	68.7	0.52	0.59	3.2	1
Iron/copper uptake	31	12.0	0.66	0.90	4.2	351
Phosphoglycerides biosynthesis	30	37.8	0.65	0.60	2.9	22
Zinc starvation	8	37.8	0.59	0.88	4.6	3
Hexose transporters	12	34.5	0.65	0.53	3.2	16
Galactose utilization	16	20.6	0.54	0.81	4.0	915
Mid sporulation	101	10.8	0.79	0.70	2.6	4158
Meiosis	41	11.6	0.70	0.44	3.5	1464
Mating type a signaling genes	18	18.4	0.44	0.44	8.0	17
Mating	111	38.0	0.72	0.79	2.7	22673
Mating type α signaling genes	18	19.4	0.56	0.89	3.8	1
Phosphate utilization	29	24.2	0.75	0.76	3.2	6528
Glycolysis	20	27.5	0.52	0.90	3.7	84
Ergosterol biosynthesis	30	27.5	0.79	0.77	3.1	283
Histones	25	36.2	0.54	0.48	3.2	1286
Cell cycle G1/S	82	43.8	0.65	0.90	3.6	1717
Cell wall (bud emergence)	17	45.8	0.60	0.89	4.0	67
Cell cycle M/G1	29	29.1	0.60	0.97	3.9	1747
Cell cycle G2/M	30	27.0	0.67	0.90	3.6	2787
Uracil synthesis/permeases	8	10.9	0.64	0.88	3.5	19
Fatty acid synthesis++	23	51.3	0.81	0.48	3.1	4
Ribosomal proteins	107	56.1	0.74	0.88	3.3	20633
rRNA processing	80	51.6	0.61	0.40	2.7	45515

Table SI. 40 of the modules found by PISA that we could assign a name to. For each module we list the number of genes in the module, the number of conditions that had a significant contribution to the module, how consistent the module was from each run to the next, the maximal overlap with a module found by ISA (using 200,000 seeds at each threshold from 1.8 to 15.0), the threshold value t_G at which that overlap was found, and how many times such an ISA module was found.