JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 7, 2005

Pp. 909–927

Probabilistic Discovery of Overlapping Cellular Processes and Their Regulation

ALEXIS BATTLE,[1] ERAN SEGAL,[2] and DAPHNE KOLLER[1]

ABSTRACT

In this paper, we explore modeling overlapping biological processes. We discuss a probabilistic model of overlapping biological processes, gene membership in those processes, and an addition to that model that identifies regulatory mechanisms controlling process activation. A key feature of our approach is that we allow genes to participate in multiple processes, thus providing a more biologically plausible model for the process of gene regulation. We present algorithms to learn each model automatically from data, using only genomewide measurements of gene expression as input. We compare our results to those obtained by other approaches and show that significant benefits can be gained by modeling both the organization of genes into overlapping cellular processes and the regulatory programs of these processes. Moreover, our method successfully grouped genes known to function together, recovered many regulatory relationships that are known in the literature, and suggested novel hypotheses regarding the regulatory role of previously uncharacterized proteins.

Key words: cellular processes, gene regulation, probabilistic relational models

1. INTRODUCTION

A LIVING CELL IS A COMPLEX SYSTEM that needs to perform various functions and adapt to changing environments. Essential to the cell's ability to respond to different circumstances is its organization into a set of *processes*, whose activity varies according to circumstance. The activation of these processes is often controlled by a variety of regulatory signals, which are themselves triggered by various aspects of the current state. Genomewide measurements of mRNA expression level across multiple experimental conditions provide us with a global picture of the cell's activities and provide the potential for a high-level understanding of its behavior.

The most common approach for analyzing gene expression is to *cluster* genes whose expression profile is similar over a range of experimental conditions (Eisen *et al.*, 1998; Cheng and Church, 2000). As the clusters correspond to sets of genes that respond similarly in different circumstances, their member genes are likely to share a common function. However, these approaches group genes into mutually exclusive

[1]Computer Science Department, Stanford University, Stanford, CA 94305-9010.
[2]Center for Studies in Physics in Biology, The Rockefellar University, New York, NY 10021.

clusters, whereas the real biological system is much more intricate, with many genes playing multiple roles in different circumstances.

To address this issue, other approaches (Segal *et al.*, 2001; Tanay *et al.*, 2002; Ihmels *et al.*, 2002; Lazzeroni and Owen, 1999; Alter *et al.*, 2000) have been proposed for discovering *processes*, or overlapping groups of genes, thereby providing a more realistic model for cellular activity. These approaches vary in how the different processes combine to explain the expression data. Some (Tanay *et al.*, 2002; Ihmels *et al.*, 2002) do not attempt to provide a single model of the entire expression matrix, but rather look only for significant submatrices. In these approaches, a single gene expression measurement can be associated with multiple processes, and there is no attempt to explain how the actual expression value arises from the combination of the gene's module assignments. In others (Segal *et al.*, 2001), a gene's module assignment can vary from one context to another, but in each experiment, a gene belongs to precisely one module. This type of model fails to capture situations where the same gene can play a role within multiple processes simultaneously. Finally, some approaches (Lazzeroni and Owen, 1999; Alter *et al.*, 2000) decompose the gene expression matrix as a sum of overlapping "layers," so that each expression value is a sum of components associated with the different processes to which the gene belongs.

In this paper, we propose an approach that also assigns genes to multiple processes. In this framework, which we call the *overlapping process* (*OP*) model, we assume that a process is active to different degrees in different conditions. Specifically, each process p has some activity level $a.C_p$ in each given array a. The expression level of a gene in an array a is a sum of the activities of the processes with which it is associated. Thus, like SVD (Alter *et al.*, 2000) and the Plaid approach (Lazzeroni and Owen, 1999), the OP model decomposes the expression matrix as a sum of layers. However, our definition of processes differs significantly, in that we incorporate an explicit bias against having genes that participate in a large number of processes. This assumption is a better fit with current biological understanding, and the resulting learned models contain processes that are biologically more coherent. (See Section 2 for further discussion.)

However, finding the genes that participate in each process provides only a partial view of the biological system. In many cases, we are also interested in discovering the regulatory mechanism that controls the activation of each process and its response to different circumstances. This type of analysis is typically done as a postprocessing step, by searching, for example, for common *cis*-regulatory motifs in the promoter region of the genes in the process (Wingender *et al.*, 2001), or (more recently) for correlation with protein–DNA binding data (Lee *et al.*, 2002). However, such approaches are limited both by the amount of noise in these data, and, more importantly, by the fact that many regulatory relationships are *context specific*, occurring only under certain conditions, a phenomenon which does not manifest in these data.

Recent work (Pe'er *et al.*, 2001; Segal *et al.*, 2003b) shows that regulatory relationships can be learned directly from gene expression data, in a way that accounts for the context-specificity of regulatory events. Moreover, as shown by Segal *et al.* (2003b, 2003a), by searching for sets of co-regulated genes and their actual regulatory program simultaneously, a much more accurate organization of genes into processes can be obtained. However, while their approach models aspects relating to regulatory mechanisms, they partition the genes into mutually exclusive processes, and thus their models are limited in the comprehensiveness of the regulatory picture they provide.

We build on the work of Segal *et al.* (2003b), extending our basic model of overlapping processes to incorporate rich models of gene regulation. Our overall framework, which we call the coregulated overlapping processes model, or *COPR* model (pronounced "copper"), explicitly models both the assignment of genes to multiple overlapping processes and the regulatory program associated with each process. It provides a unified probabilistic framework of gene regulation for multiple overlapping biological processes.

Our model makes the simplifying assumption that regulation is done at the level of processes, rather than individual genes. Thus, we assume that each process is associated with some (unknown) regulatory program, which determines its activity level $a.C_p$ in each array a. Thus, the activity of p is different in different arrays, but it is governed by the same set of rules. The process's activity level is assumed to be a function of its regulators. We use a regulatory program that captures two of the most important forms of regulation: both the combinatorial (context-specific) nature of some types of gene regulation and the dependence of the activity level of a process on the actual abundance of the regulatory protein (as indicated by its mRNA level) in the cell.

We present an efficient algorithm for learning a COPR model automatically from data, using only the raw gene expression measurements as input. In our learning task, we must associate genes with processes,

as well as learn the regulatory program of each process—both the set of regulators for each process and the program itself. This learning problem is quite challenging, as both the (discrete) assignment of genes to processes and the (continuous) activity levels of processes in arrays are all unobserved.

We evaluate our approach, examining both a model that includes reguatory programs and a model that does not include regulation. We test our model on two expression datasets, including a large compendium of 394 yeast microarrays compiled from four different studies. We evaluate its performance relative to a variety of existing biological data sources, including known gene annotations (Ashburner, 2000), presence of motifs in the gene's promoter regions (Wingender *et al.*, 2001), and correlation with protein–DNA binding data (Lee *et al.*, 2002). We show that (even without the regulatory model) our more biologically plausible model of overlapping processes results in much "cleaner" and more biologically coherent processes than previous approaches.

For our richer COPR model, we show that, relative to all of these biological data sources, our approach discovers groups of genes that correspond to biologically coherent processes, significantly outperforming the module network framework of Segal *et al.* (2003b). Notably, the regulatory models also lead to significant improvements over the simple OP model, which does not include gene regulation. We also evaluate the predicted activity levels for our learned processes and show that they are very consistent with their role in the cell. Overall, our results support our basic assumptions, that overlapping processes provide a better model of cellular activity than disjoint gene clusters and that by searching simultaneously for co-regulated genes and for their regulatory programs, we converge to models that are biologically more plausible.

2. RELATED WORK

As described above, our framework addresses two main goals: placing genes into functional groups and discovering regulatory mechanisms for those gene groups. The first goal, grouping genes into meaningful groups, is achieved by the simple OP model, and is shared by approaches such as clustering (e.g., Eisen *et al.* [1998]). Standard clustering partitions genes into mutually exclusive groups, while our algorithm places genes in multiple processes simultaneously.

Another approach to relaxing the assumption of mutually exclusive clusters is "soft clustering," which assigns probabilities that each member belongs to each one group. This type of model is more reflective of uncertainty about the cluster assignment of a gene, rather than its assignment to multiple clusters simultaneously. Other approaches, such as biclustering (Cheng and Church, 2000) or context-specific clustering (Segal *et al.*, 2001) allow genes to participate in different groups in different contexts, but in any given array, a gene still belongs to exactly one group. In other words, each individual expression measurement is explained by a gene's membership in precisely one cluster. In the COPR model, a gene can belong to multiple processes simultaneously, and a gene's expression value in any condition is a sum of the activity of all of the processes in which the gene participates.

Thus, our model decomposes the expression matrix as a sum of process activity matrices. This characteristic makes COPR more similar to singular value decomposition (SVD) (Alter *et al.*, 2000) or the Plaid model of Lazzeroni and Owen (1999). These approaches also decompose the expression matrix as a sum of activities of (overlapping) *components* or *layers*, which are roughly analogous to our processes. However, our definition of processes differs significantly, in that we incorporate an explicit bias against having genes that participate in a large number of processes. Thus, in our learned COPR models, each gene tends to belong to at most two or three processes. By contrast, SVD or Plaid often produce models where genes belong to a very large number of processes. As we show, the sparser models are more biologically plausible and contain processes that are more biologically coherent.

Finally, and most notably, none of these approaches include a model of gene regulation. Including gene regulation in our COPR model allows us both to discover regulatory relationships and to use these discovered relationships to actually improve the grouping of genes. The COPR model presented in this paper is based on the regulatory framework proposed in the module network framework (Segal *et al.*, 2003b), but extends it in two significant ways. First, it allows the use of overlapping processes, whereas the module network framework assumes disjoint clusters. Second, it extends the notion of a regulatory program to allow both combinatorial and linear dependence on the abundance of regulators.

3. PROBABILISTIC MODEL

The COPR model is based on the language of *probabilistic relational models* (PRMs) (Koller and Pfeffer, 1998; Friedman *et al.*, 1999; Segal *et al.*, 2001), which represents the domain in terms of the different biological entities that interact in it: genes, arrays, expression measurements, regulators, and biological processes. Each object is also associated with a set of attributes that are relevant to its interactions with the other entities.

3.1. Gene expression model

The first component of the COPR (and simpler OP) model represents the gene expression and its decomposition into the activity level of processes. This component includes a set $\mathbf{G}$ of n gene objects, $\mathbf{G} = \{g_1, \ldots, g_n\}$, a set $\mathbf{A}$ of k array objects, $\mathbf{A} = \{a_1, \ldots, a_k\}$, and a set $\mathbf{E}$ of expression objects $\mathbf{E} = \{e_{1,1}, \ldots, e_{n,k}\}$, one for each gene in each array. Each expression object e is associated with a gene object $e.Gene = g$, an array object $e.Array = a$, and a real-valued attribute $e.Level$ denoting the mRNA expression level of $e.Gene = g$ in $e.Array = a$.

As we discussed, a key property of our approach is that it allows genes to participate in multiple processes. To this end, COPR explicitly includes a set of j *processes*; each gene object g is associated with a set of binary *process membership* attributes $g.M_1, \ldots, g.M_j$, where $g.M_p$ denotes whether g is a member of process p. As processes may be active to various degrees in different arrays, we also define a set of continuous *activity level attributes* $a.C_1, \ldots, a.C_j$ for each array a, where $a.C_p$ corresponds to the degree to which process p is active in array a.

The expression level of gene g in array a is assumed to be a sum of g's expression levels in each of the processes in which it participates, where g's expression level in process p is the activity of the process $a.C_p$. More precisely, let $g.\mathbf{M}$ be the set of all g's membership variables, and $a.\mathbf{C}$ be the set of all a's activity level variables. We assume that $e.Level$ is normally distributed with mean that is equal to the sum, over processes p in which which g participates, of the activity level of p:

$$P(e.Level \mid g.\mathbf{M}, a.\mathbf{C}) = \mathcal{N}\left(\sum_{p=1}^{j} g.M_p \cdot a.C_p; \sigma_a^2\right) \tag{1}$$

where σ_a^2 is the variance associated with array a.

3.2. Regulatory model

In our OP model, the activity level of a process p is simply a stand-alone variable, with a distinct value in each array. In our richer models, where we try to encode regulatory interactions, we assume that the activity levels are themselves determined by some regulatory program. Specifically, we assume that genes in the same process are co-regulated and therefore share the same regulatory mechanism. Thus, we define a *regulation program* for each process, which is then shared across all genes assigned to that process. To model regulation programs, we extend the framework of Segal *et al.* (2003b), where each regulation program is defined in terms of some set of regulators and describes the expression of genes in the process as a function of the mRNA expression level of these regulators. We therefore assume that the model contains a set of t *candidate regulators*, where for each regulator r and each array a we have a continuous attribute $a.R_r$ that denotes the mRNA expression level of regulator R_r in array a. We define $a.\mathbf{R} = a.R_1, \ldots, a.R_t$.

We model the activity level of process p in the different arrays as a stochastic function of the expression level of some small set of process regulators. The regulators are specified separately for each process, allowing us to represent regulation programs that are specific to different processes. Thus, for each process p, the COPR model specifies a set of d parent regulator attributes $\mathcal{R}p = \{R_{p,1}, \ldots, R_{p,d}\}$, and the activity level of process p in array a, $a.C_p$, is then described as a stochastic function of $a.R_{p,1}, \ldots, a.R_{p,d}$. In our framework, this stochastic function takes the form of a *regression tree* (Breiman *et al.*, 1984). A regression tree allows us to model two types of dependence on regulators: linear dependencies, which are very common, and combinatorial regulation, which is crucial for any realistic model of gene regulation in higher eukaryotes.

As illustrated in Fig. 1(a), a regression tree $\mathcal{S}_p$ for a process p with parents $\mathcal{R}p$, is a rooted binary tree with interior tree nodes and leaf tree nodes. Each interior tree node is labeled with a *test* over one of the parents, such as $a.R_{p,1} > .7$. Each leaf node specifies a distribution over $a.C_p$. For a particular array a in which we observe the expression levels of the regulators, we define $P(a.C_p \mid a.R_{p,1}, \ldots, a.R_{p,d})$ by traversing the tree from the root and taking the appropriate branch at each interior node: the right branch if the values $a.R_{p,1}, \ldots, a.R_{p,d}$ pass the test at that node, and the left branch otherwise. We continue this traversal until we reach a leaf node ℓ. The distribution $P(a.C_p \mid a.R_{p,1}, \ldots, a.R_{p,d})$ is then specified by the distribution associated with ℓ. The leaf distribution at each leaf ℓ in is a *linear Gaussian*—a Gaussian whose mean is a linear function of a subset of *linear parents* $\mathcal{R}_\ell = \{R_{\ell,1}, \ldots, R_{\ell,f_\ell}\}$ of the parent regulators $\mathcal{R}_p$. Thus, if array a reaches leaf ℓ in the tree, it will have a Gaussian distribution $\mathcal{N}(\mu_a; \sigma_\ell^2)$, where $\mu_a = b_{\ell,0} + \sum_{i=1}^{f_\ell} b_{\ell,i} \cdot a.R_{\ell,i}$, $b_{\ell,i}$ is the weight of regulator i in the leaf distribution of leaf ℓ, and σ_ℓ^2 is the variance associated with the leaf. Thus, we represent the activity levels both by combinatorial interactions of regulators and by a linear function of regulator expression levels. We note that this regulation model is an extension of the model of Segal *et al.* (2003b), which incorporated only combinatorial regulation and did not allow linear dependence on the actual expression level of the regulators.

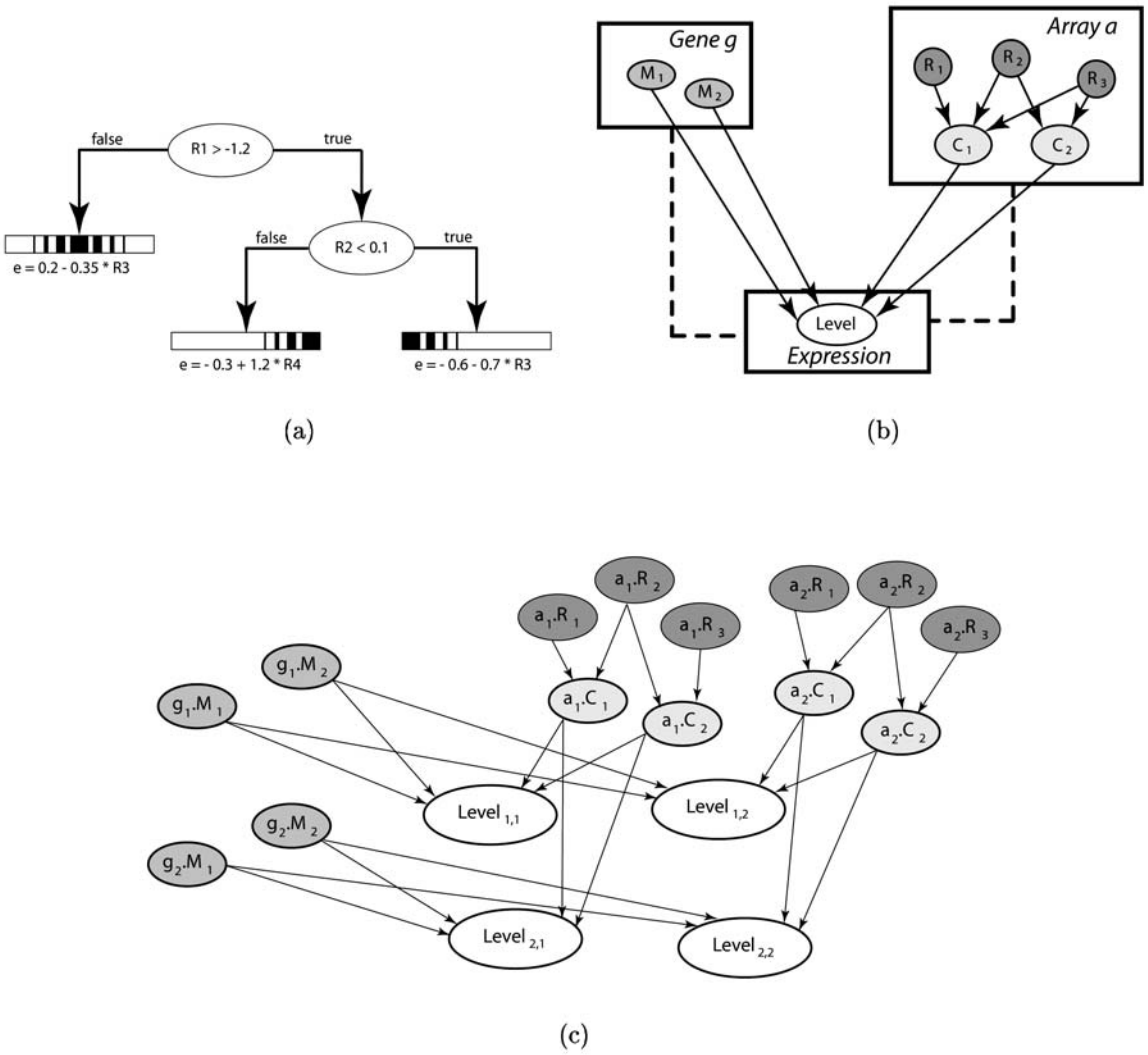

FIG. 1. (**a**) A sample regression tree predicting a different distribution over the activity level of a process, depending on regulator levels. (**b**) COPR PRM with two processes and three regulators. (**c**) A simple instantiation of the COPR PRM in (b) for two genes and two experiments.

3.3. Model summary

Our overall COPR PRM model puts together these two components. The probability of the expression levels **E**.*Level* given the gene process membership variables **G**.**M** and the process activity level **A**.**C** is as described in our gene expression model. The probability of **A**.**C** given the regulators is as described in the regulatory model. To complete the description of COPR, we define the gene membership attributes, $g.\mathbf{M}$, to have no parents; we simply associate with each process p a prior distribution over the membership of genes in this process $P(g.M_p) = q_p$. The regulator expression levels $a.R_1, \ldots, a.R_t$ also have no parents. We assign a Gaussian distribution $\mathcal{N}(\mu_r; \sigma_r^2)$, noting that, as these regulator attributes are always observed, this distribution will not play a role when we learn the model from data.

A simple example of COPR PRM for the case of two processes and three regulators is shown Fig. 1(a). For any set of genes and arrays, the model defines a probability distribution over the gene process memberships, the array activity levels, and the expression levels of genes in the arrays. This probability distribution is defined as a Bayesian network, whose structure and parameterization is determined by the model. An example of the instantiation of the COPR model, for the case of two genes, two arrays, two processes, and three regulators, is shown in Fig. 1(b). The joint distribution defined by our instantiation can be written in terms of the objects, their attributes and the distributions specified above:

$$\begin{aligned} P(\mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathbf{A}.\mathbf{R}, \mathbf{E}.\mathit{Level}) = & \prod_{p=1}^{j} \prod_{g \in \mathbf{G}} P(g.M_p) \\ & \cdot \prod_{a \in \mathbf{A}} P(a.C_p \mid a.R_{i_1}, \ldots, a.R_{i_d}) P(a.R_{i_1}, \ldots, a.R_{i_d}) \\ & \cdot \prod_{e \in \mathbf{E}} P(e.\mathit{Level} \mid e.\mathit{Gene}.\mathbf{M}, e.\mathit{Array}.\mathbf{C}). \end{aligned} \tag{2}$$

In the OP model, we simply eliminate the terms $P(a.C_p \mid a.R_{i_1}, \ldots, a.R_{i_d})$ from the probabilistic model. Note that, in principle, one could introduce a prior over the variable $a.C_p$, which could be learned from data. We chose to simplify this model and fixed this prior distribution to be uniform.

In addition to modeling multifunctionality of genes and complex regulatory mechanisms, we can easily obtain regulatory information directly from the COPR model: The regulators of each process p are specified by the sets $\mathcal{R}p$, and the regression tree for process p tells us exactly how each regulator affects the activity level of the genes participating in process p.

4. LEARNING

Learning a COPR model from microarray data alone is a complex task. Microarray data provides the values for each $e.\mathit{Level}$ variable and the values for each regulator expression variable $a.R_{p,i}$. All of the assignments of genes to processes are hidden, as are all activity levels. In addition, we do not know which regulators $\mathcal{R}p$ control each process p, or by what program the regulators control their target processes. The parameters q_p, specifying the probability of a gene belonging to process p, must also be learned. Thus, from expression data alone, the goal is to group genes into processes, to estimate process activity levels for each experiment, and to learn the regulatory control programs governing the activity of each process.

Overall, this problem can be viewed as one of learning a probabilistic model—structure as well as parameters—from partially observed data. We address this problem using a hard-assignment variant of the *structural EM (SEM)* algorithm of Friedman (1998). Like the EM algorithm of Dempster *et al.* (1977), SEM iterates over two steps: In the *E-step*, it finds a "completion" of the values to the hidden variables given a current model; in the *M-step* it re-estimates the model given our current "completion" of the data. In SEM, the model structure as well as the parameters are both learned in the M-step.

We now describe each of the main steps of our algorithm and then describe how we put them together in a single learning algorithm.

4.1. E-step

We first consider the problem of finding the most likely joint assignment to **G**.**M** and **A**.**C**. This task is a hard one: As can be seen in Fig. 1(c), these variables are all correlated via their joint influence on the expression levels. For example, consider two genes g and h; h's assignment to processes influences our estimates of the $a.C_p$ variables, which in turn influence our membership probabilities for g. Due to these dependencies, the exact computation of the E-step is intractable for large domains. The presence of a large number of correlated hidden variables makes the inference task intractable for large models. The problem is further complicated by the fact that it involves both discrete and continuous variables, a setting where even very simple network structures are intractable (Lerner and Parr, 2001).

However, our model is such that if we were given the values of $a.C_p$ for all a, p, then the assignments of the different genes to processes are rendered independent. Likewise, given a fixed assignment of genes to processes ($g.M_p$), the activity levels for each array a can be estimated from these assignments and from the expression data in a alone, without knowing the activity levels in other arrays.

This key observation allows us to use a very effective form of local search algorithm in performing the E-step. We decompose the E-step into two substeps: finding the most likely assignment of genes to processes, given the process activity levels, and finding the most likely assignment of process activity levels given gene assignments. The independence properties of the network make each of these subtasks considerably easier.

Specifically, starting from an initial assignment of genes to processes (which could come from standard clustering methods), we find the most likely activity levels **A**.**C**. We then fix these activity levels and find the most likely assignment to **G**.**M**. Each step increases the joint likelihood $P(\mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathbf{E}.\mathit{Level})$ given the current parameters, and thus the process is guaranteed to converge. The resulting assignment to these variables is a fairly strong local maximum: No step that adapts only the gene memberships or the array activity levels can improve the likelihood; however, a step that adapts both gene memberships and array activities might.

4.1.1. Assigning genes to processes. The first of these two substeps—finding the assignment of genes to processes—can be formulated as follows: given an assignment to **A**.**C** and a model $\mathcal{M}$, optimize

$$\begin{aligned}&\operatorname{argmax}_{\mathbf{G}.\mathbf{M}} P(\mathbf{G}.\mathbf{M} \mid \mathcal{M}) P(\mathbf{A}.\mathbf{R} \mid \mathcal{M}) P(\mathbf{E}.\mathit{Level} \mid \mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathcal{M}) P(\mathbf{A}.\mathbf{C} \mid \mathbf{A}.\mathbf{R}, \mathcal{M})\\ &\quad = \operatorname{argmax}_{\mathbf{G}.\mathbf{M}} P(\mathbf{G}.\mathbf{M} \mid \mathcal{M}) P(\mathbf{E}.\mathit{Level} \mid \mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathcal{M}). \qquad (3)\end{aligned}$$

Note that, as the activation levels are given, the presence of the regulatory model (if one is present) is irrelevant.

A key observation is that, with all activity levels given, assignments of genes to processes are independent across genes and we can find the most likely assignment for each gene separately. To perform this computation, we thus maximize $P(g.\mathbf{M} \mid \mathbf{E}_g, \mathbf{A}.\mathbf{C})$ separately for each gene g, where $\mathbf{E}_g$ is the row in the expression matrix corresponding to the gene g. More precisely, this computation can be done as follows:

$$\mathbf{v}^* = \operatorname{argmax}_{\mathbf{v}'} P(g.\mathbf{M} = \mathbf{v}' \mid \mathbf{E}_g, \mathbf{A}.\mathbf{C}), \qquad (4)$$

where

$$P(g.\mathbf{M} = \mathbf{v} \mid \mathbf{E}_g, \mathbf{A}.\mathbf{C}) = \alpha \prod_p P(g.M_p = v_p) \prod_{e \in \mathbf{E}_g} P(e.\mathit{Level} \mid g.\mathbf{M} = \mathbf{v}, \mathbf{A}.\mathbf{C}),$$

where $v_p \in \{0, 1\}$ and α is a normalization constant. The expression inside the final term in the product is simply the Gaussian model for the expression level given its parents.

For models that include a large number of processes, we cannot perform this maximization over $g.\mathbf{M}$ exactly. The number of calculations required for each gene is exponential in the number of processes, since every possible joint assignment to $g.\mathbf{M}$ must be considered. In these cases, we use an approximation. Instead of considering every possible assignment to $g.\mathbf{M}$, we *include* only a subset $g.\mathbf{M}_I$ of processes, and *exclude* all others $g.\mathbf{M}_E$, forcing their value to 0. To select our subset, we relax the problem and allow each $g.M_p$ to be any real value between 0 and 1. We then maximize (4) subject to this relaxation.

This problem reduces to a bounded least squares problem, which we can solve exactly (Bjorck, 1996). We then select $g.\mathbf{M}_I$ from $g.\mathbf{M}$, by choosing those variables whose relaxed assignments are closest to 1 (in practice the majority of the variables are assigned to 0 in the relaxed solution). Finally, we find the most likely $\{0, 1\}$ assignment to $g.\mathbf{M}_I$ with all other variables fixed to 0 by maximizing

$$P(g.\mathbf{M}_I = \mathbf{v} \mid \mathbf{E}_g, \mathbf{A}.\mathbf{C}, g.\mathbf{M}_E = \mathbf{0})$$
$$= \alpha \prod_{p \in I} P(g.M_p = v_p) \prod_{e \in \mathbf{E}_g} P(e.Level \mid g.\mathbf{M}_I = \mathbf{v}, g.\mathbf{M}_E = \mathbf{0}, \mathbf{A}.\mathbf{C}).$$

In our implementation, we experimentally selected the threshold for including a process p in $g.\mathbf{M}_I$. In practice, we found that a large number of the continuous memberships were actually assigned to 0. Once each gene has been assigned a candidate set of processes, the remaining discrete optimization problem is then to be solved exactly using exhaustive enumeration.

4.1.2. Computing activity levels. In the second part of the E-step, we compute the most likely assignment to the activity levels $\mathbf{A}.\mathbf{C}$ with the regulatory program and gene memberships $\mathbf{G}.\mathbf{M}$ fixed. Our goal here is to maximize

$$\begin{aligned}&\text{argmax}_{\mathbf{A}.\mathbf{C}} P(\mathbf{G}.\mathbf{M} \mid \mathcal{M}) P(\mathbf{A}.\mathbf{R} \mid \mathcal{M}) P(\mathbf{E}.Level \mid \mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathcal{M}) P(\mathbf{A}.\mathbf{C} \mid \mathbf{A}.\mathbf{R}, \mathcal{M}) \\ &= \text{argmax}_{\mathbf{A}.\mathbf{C}} P(\mathbf{A}.\mathbf{C} \mid \mathbf{A}.\mathbf{R}, \mathcal{M}) P(\mathbf{E}.Level \mid \mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathcal{M}). \end{aligned} \quad (5)$$

For any array a and each process p, the regulator variables are all observed. Thus, we know precisely which path is followed down the regression tree for p and the values of the linear parents at the corresponding leaf. Thus, $\mathcal{M}$ and the values of the regulator variables define a Gaussian distribution $\mathcal{N}(\mu_{p,a}; \sigma_p^2)$ over the values of $a.C_p$. Given the gene process assignments, the variables $a.C_p$ for different arrays a are conditionally independent. Thus, for each a, we need to optimize

$$\begin{aligned}&\text{argmax}_{a.\mathbf{C}} \sum_{e \in \mathbf{E}_a} \log\left(\frac{1}{\sqrt{2\pi}\sigma_a} \exp\left(\frac{(e.Level - \mu_e)^2}{2\sigma_a^2}\right)\right) + \left(\sum_p \log\left(\frac{1}{\sqrt{2\pi}\sigma_p} \exp\left(\frac{(a.C_p - \mu_{p,a})^2}{2\sigma_p^2}\right)\right)\right) \\ &= \text{argmin}_{a.\mathbf{C}} \frac{1}{2} \sum_{e \in \mathbf{E}_a} (e.Level - \mu_e)^2 + \frac{\sigma_a^2}{2\sigma_p^2} \sum_p (a.C_p - \mu_{p,a})^2, \end{aligned} \quad (6)$$

where $\mathbf{E}_a$ are all of the expression measurements associated with the array a and for each $e \in \mathbf{E}_a$, μ_e is the mean of the Gaussian over the expression level, as specified in (1). Note that μ_e is a linear function of the values $a.C_p$, whose coefficients are determined by the (known values of) $g.M_p$. The form of (6) allows us to use a simple least squares computation to solve this minimization problem optimally and efficiently. Note that, in the OP model, the second term in (6) disappears.

4.2. M-step

In the M-step, we are given a complete assignment to the hidden variables, and our task is to learn a good model. In the case where we have no regulatory programs, this learning step is trivial: The only model parameters are the variances σ_a^2 of the expression measurements for array a and the probabilities q_p which encode the probability of gene membership in process p. These parameters are estimated using maximum likelihood, based on the given hard assignments to the hidden variables. Specifically, the probability q_p is estimated as the fraction of genes assigned to process p in the current hard assignment, and the variances σ_a^2 associated with (1) are also assigned to be their empirical estimates.

In richer models, incorporating a regulatory program for processes, the learning task is much more complex, involving both structure and a richer set of parameters. The structure $\mathcal{S}$ specifies the structure $\mathcal{S}_p$ of the regression tree for each process p and the set of linear parents $\mathcal{R}\ell$ at each leaf ℓ in the tree $\mathcal{S}_p$. The parameters specify the linear coefficients and the variances in the regression tree, the variances σ_a^2 of the expression measurements for array a, and the probability q_p of gene membership to process p.

To select $\mathcal{S}$, we use a Bayesian model selection approach, which explicitly accounts for our uncertainty about the parameters using probability distributions over the values of $\boldsymbol{\theta}$. We then search for a model that has high posterior probability given the expression data $\mathcal{D}$, integrating over all possible values of $\boldsymbol{\theta}$: $P(\mathcal{S} \mid \mathcal{D}) \propto P(\mathcal{S})P(\mathcal{D} \mid \mathcal{S}) = P(\mathcal{S}) \int P(\mathcal{D} \mid \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}) d\boldsymbol{\theta}_{\mathcal{S}}$. We choose to use a structure prior $P(\mathcal{S})$that penalizes the number of regulators in each regulatory program. Specifically, our prior shrinks exponentially with the number of distinct regulators. Our Bayesian score is simply the logarithm of this posterior probability.

We search for a set of regression trees $\mathcal{S}_p$ that maximize this posterior probability, given a hard assignment to the hidden variables. Given a fixed assignment to $\mathbf{A}.\mathbf{C}$, the optimization task for each process p is independent and depends only on the activity levels of p in each array, $Array.C_p$. Thus, our task for process p reduces to one of finding the regression tree that optimizes the Bayesian score, given the data $\mathbf{A}.C_p$.

To find a high-scoring regression tree $\mathcal{S}_p$, we perform a greedy search over possible structures. We begin with a single root node in the tree, which would predict the activity level of process p in all conditions. Then, we consider every possible single split that could be added to the tree. A split is a test of the form $a.R_i > z$ for any regulator expression $a.R_i$ and any real value z. Adding this split to the tree would allow the model to specify two distributions, one for activity levels in experiments where $a.R_i > z$ and one for experiments where $a.R_i \leq z$. Note that, while the addition of a split can never lower a likelihood score, it can hurt the Bayesian score. We calculate the change in score for each possible split, and if any improve the score, we use the best split to modify the tree. When no more splits can improve the score, the tree structure and split cutoff values are fixed.

To score each split, we first note that the Bayesian score of a regression tree can be decomposed as a sum over terms, each reflecting the score of a single leaf node in the tree. Thus, to calculate the improvement made by a split, we need only consider the contribution to the score of the affected leaves. Scoring a leaf, unfortunately, is not trivial. Recall that each leaf ℓ defines a linear Gaussian distributions that depends on some set of linear parents $\mathcal{R}\ell$. Given a fixed set of linear parents, we can compute the Bayesian score of the leaf by integrating over all possible values of the linear coefficients (with a Gaussian prior), as described by Geiger and Heckerman (2002). We select the set of linear parents at each leaf using a greedy search, adding linear parents which improve the Bayesian score of the leaf. We note that this computation is performed every time a new leaf is considered as part of a split operation.

The result of this computation is a model structure specifying the regression tree structure $\mathcal{S}_p$ for every process p. The regression tree parameters are now computed as follows: the variance σ_ℓ^2 associated with the leaf ℓ is estimated using its empirical estimate, and the linear coefficients at each leaf are estimated using standard linear regression.

4.3. Algorithm summary

We initialize the algorithm by using a standard expression clustering technique to select an assignment of genes to processes. The result is a model where processes are disjoint and each gene belongs to precisely one process. We then find an initial set of activity levels using the least squares method, using a degenerate regression tree (with only a root node) as the regulatory model.

With this initialization, the algorithm repeatedly executes an M-step and an E-step, as specified above. At each iteration, a new regulatory model is learned using the current hard assignments to $\mathbf{G}.\mathbf{M}$ and $\mathbf{A}.\mathbf{C}$. This regulatory model is then used to re-estimate new values for the activity levels $\mathbf{A}.\mathbf{C}$ and of the gene process memberships $\mathbf{G}.\mathbf{M}$.[1] The E-step and M-step are repeated until the gene process assignments stabilize. At that point, no further changes can occur, and the algorithm has converged. We then have a complete COPR model, including an assignment of genes to processes, an estimate of the activity level for each process in each array, and a learned regulatory mechanisms for each process.

An important feature of our algorithm is that the membership and activity of each process are not learned in isolation. Rather, our model is learned over all processes simultaneously, allowing information

[1]Note that each of these two substeps is executed only once before the next M-step; as these two substeps do not achieve a global optimum for the E-step, we could also iterate the substeps of the E-step until convergence, or for some number of steps. Our experiments suggest that this choice makes very little difference.

and (probabilistic) conclusions from one process to propagate and influence our conclusions about another. For instance, assume that our learning process places a gene g into process p at some step and that this membership explains g's expression data very accurately. In this case, g will be less likely to be a member of other processes, allowing other genes assigned to the process to have a stronger influence on the activity level profile of the process.

5. EXPERIMENTAL RESULTS

We evaluated the performance of our models on yeast gene expression data. Our analyses included examining statistical properties, evaluating our learned COPR models with known biological attributes of genes and processes, and comparing both the simple OP model and the richer COPR model to other models on large expression datasets. In each case, we applied the learning algorithm described in Section 4, specifying a candidate set of regulators and a number of processes.

5.1. The overlapping process model

First, we examined the statistical properties of the OP model, which does not include any regulatory programs for the processes.

5.1.1. Synthetic data. We began by evaluating the OP model on synthetic data. These experiments test whether we recover structure known to be present in the data. We generated a synthetic dataset by sampling from a given OP model. To make the data realistic, we used OP models learned from real biological data (Gasch *et al.*, 2000). Specifically, we first learned an OP model with seven processes. We then sampled data for 500 genes and 173 experiments (the original data contained 173 experiments) from the model: assignments of genes to processes were sampled from the distribution our model had for the $\mathbf{G}.\mathbf{M}$ variables, and expression data was then derived by computing the expected expression levels (according to our model of expression) from the sampled assignment of genes to layers and the $a.\mathbf{C}$ means which were part of the learned model.

We then hid the true assignments of genes to processes and activity levels in arrays, as well as the original model parameters q_p, and learned a model with seven processes from the synthetic expression data using the algorithm described in Section 4. To test the robustness of our learning algorithm to noise, we also learned models using various levels of perturbations, where a perturbation level of π corresponds to shuffling π% of the expression data across all genes and experiments. To gain statistical confidence, we generated five datasets for each perturbation π and learned a model from each one.

All models were evaluated by their ability to recover the "true" assignments of genes to processes (the true assignments are the assignments in the sampled data) by performing a pairwise consistency test: we extracted all gene pairs appearing in the same process in our learned model and computed the fraction of these pairs appearing in the true data. We also tested the reverse, extracting all the true pairs and computing the fraction of these pairs appearing in a learned model. The results are summarized in Fig. 2(a), indicating that our algorithm reconstructs the true structure with very high accuracy even if 30% of the data is perturbed: gene pairs that are assigned to the same process in the true data are likely to appear in our learned model and vice versa. Note that in fully randomized data (100% perturbation), a high fraction of the pairs in the learned model were indeed present in the true data (58 ± 0.4%). This occurs since the randomized data contains much weaker patterns and the total number of pairs learned is small, as can be seen by the poor coverage (21.7 ± 2.8%) of true pairs in these models.

As another evaluation, we measured the ability of our learned models to predict unseen data, by computing the likelihood that each model assigns to held out data. Specifically, we randomly partitioned the data into five equally sized sets of 100 genes and learned five models from all five possible combinations of four sets. For each such model, we computed the likelihood it assigned to the held out subset. We compared these results to the likelihood that the "true" model from which the data was sampled assigned to the held out test data. These experiments were also performed in the presence of varying levels of perturbations. The results are summarized in Fig. 2(b). As can be seen, the test set likelihood is comparable (and even better with very little noise) for up to 30% perturbations, dropping sharply as more noise is added.

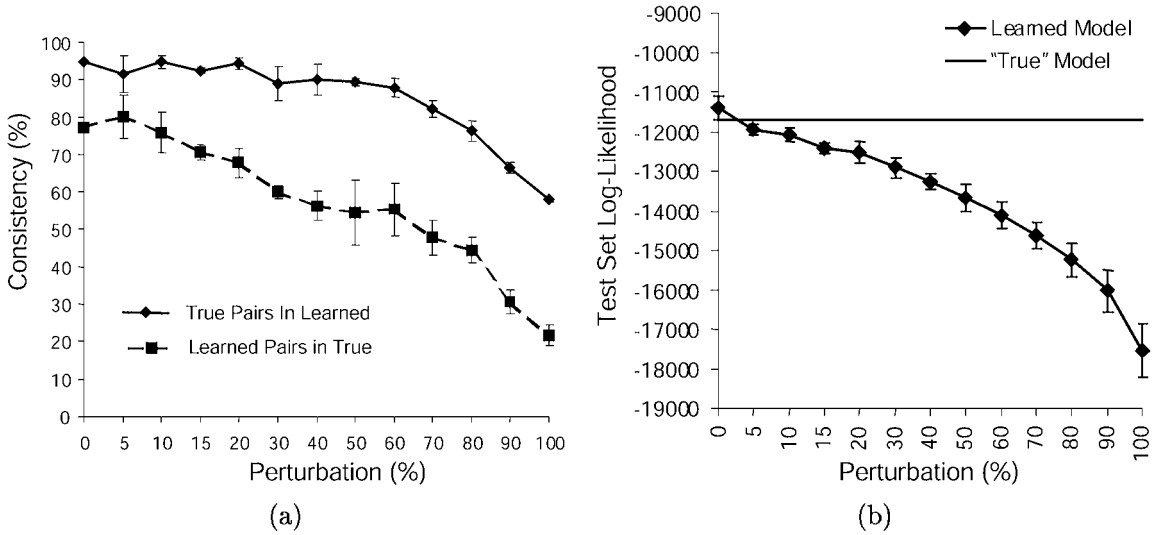

FIG. 2. Evaluation of OP model learning on synthetic data. (**a**) Fraction of learned pairs appearing in the true data and fraction of true pairs in the learned model for various levels of perturbations. (**b**) Log-likelihood on test data achieved for learned models for various levels of perturbations.

Recall that when the number of processes is large, we resort to the approximation described in Section 4. To evaluate our approximate algorithm, we learned a 12 process model, where we could apply the exact algorithm and compare the results. In this case, 77.2% of the genes had the same assignment to processes in the approximation and exact algorithm. However, the training set likelihoods of both models were practically the same, implying that the errors made by the approximation had little effect. Indeed, when comparing the test set likelihoods of both models, the differences were negligible.

5.1.2. Real data. To obtain a more comprehensive model of biological processes and regulation, we applied our method to the 173 yeast microarrays of Gasch *et al.* (2000), which measured the response of yeast to various conditions of stress. For this evaluation, we selected 1,010 genes that had significant changes in gene expression (eliminating the ESR genes for which clustering is trivial), and the full set of 173 arrays, and learned an OP model with 30 processes.

Overall, our model predicted that 24 genes do not participate in any process, 552 genes participate in only one process, 257 in two, 119 in three, and 58 in four or more processes. As a comparison, we also tested a Plaid model with 30 processes learned from the same data. (We obtained the Plaid software from *www-stat.stanford.edu/∼owen/plaid/*.) The Plaid model assigned many more genes to layers than our model did, with 0 genes in no processes, 1 gene in one process, 4 genes in two, 10 genes in three, and 995 genes in four or more processes. According to Plaid, almost *all* genes participate in four or more processes, a situation not supported by current biological understanding.

To evaluate whether our assignments are biologically plausible, we checked whether the genes associated with each process showed any enrichment for known annotations. To do so, we used the GO (Ashburner, 2000) and KEGG (Kyoto University Bioinformatics Center, 1995) databases which assign genes to a diverse set of functional categories and biological pathways, respectively. For each process and each annotation, we counted the number of genes from the process with that annotation and compared that to the total number of genes in our dataset with that annotation. If a process we learned indeed corresponds to known biological processes, then we expect the learned process to contain a high fraction of the genes with the corresponding annotation. For each combination of process p and annotation α, we can use the hypergeometric distribution and assign a statistical significance (p-value) measure corresponding to the probability that a randomly selected group of genes of the same size have similar enrichment for α. We performed this evaluation for our OP model processes, the layers found by the Plaid model, and clusters from a standard clustering procedure.

Overall, we discovered highly significant processes relating to a variety of cellular functions. These included oxidative phosphorylation, various transport processes, protein folding, glycolysis, lipid metabolism, amino acid metabolism, carbohydrate metabolism, protein membrane targeting, ribosomal biogenesis, and cell cycle control. Some of the stronger active processes we identified were also present as Plaid layers, but Plaid layers typically included many extraneous genes, rendering the patterns less clear. For example, neutral lipid metabolism appears as a process of 17 genes with a p-value of $1.26e-21$ in our model, while in Plaid it appeared in a layer of 317 genes with a p-value of $1.22e-5$. Also, protein folding appeared as a process of 14 genes with a p-value of $1.46e-16$ in our model, while the corresponding Plaid layer had 254 genes with a p-value of $2.65e-7$. Figure 3(a) shows a scatter plot comparing the p-value for the GO and KEGG annotations that came up. We can see that, except for a small portion of cases, the p-value achieved by our approach was always better and often much better than that achieved by Plaid. We performed a similar comparison to a standard hierarchical clustering algorithm (Eisen *et al.*, 1998), where we cut the hierarchy at 30 clusters to allow for a comparison to our models. The results are shown in Fig. 3(b), where again the majority of annotations appeared with greater significance in our model.

In addition to the assignments of genes to processes, our approach attempts to reconstruct the activity levels of each process p in each array a, as captured by the predicted value of $a.C_p$. For each process, we can thus construct a vector $\mathbf{A}.C_p$ of the activity levels of p across all arrays $a \in \mathbf{A}$. We examined these activity levels and found that they were biologically plausible for their respective processes. For instance, the process associated with protein folding (process 18) had high activity levels during heat shock and exposure to diamide and low activity levels during amino acid and nitrogen depletion, reflecting accurately the biological function of the process.

It is interesting to measure the correlation between the $\mathbf{A}.C_p$ vectors for all processes p and genes that were not included in our analysis. Due to the way in which we selected the 1010 genes for our analysis, the genes included are likely to contain only a fraction of the genes associated with each process. If our model learned activity levels that indeed correspond to activity levels of real processes, then we expect to see high correlations between some of the left out genes and our learned activity levels. Indeed, there were many such genes: 614 of correlation above 0.8 and 252 of correlation below -0.8. To test whether this phenomenon could have happened by chance, we permuted the vector of expression measurements for each gene and recomputed the correlations. This experiment shows that it is highly unlikely that our computed correlations could have resulted by chance, as the most significant correlation achieved for any of the 5,147 permuted genes was -0.32.

Interestingly, there were several cases where the learned process activity levels had high correlation to the expression of known regulators (e.g., transcription factors) not included in the analysis. The web supplement lists all regulators with high correlation (or anticorrelation) to any process. Overall, we had 37 unique regulators with correlation above 0.7, of which 10 had correlation above 0.8, and 8 unique

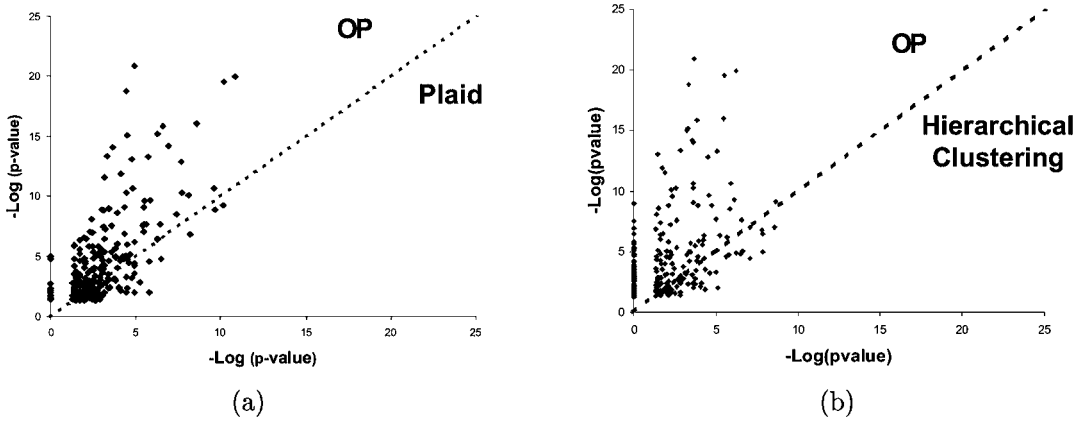

FIG. 3. Comparison of OP model to other approaches. (**a**) Scatter plot of the log p-value of different GO and KEGG annotations for layers in Plaid on the one hand (X axis) and OP model processes on the other (Y axis). (**b**) Scatter plot of the log p-value of different GO and KEGG annotations for clusters from Pearson clustering on the one hand (X axis) and OP model processes on the other (Y axis).

regulators with correlation below -0.7, of which 5 had correlation below -0.8. For 12 of the 30 processes, we learned activity levels that had extremely high correlation with known regulators. When information about the regulator was available in the literature, we could verify that the regulator was highly correlated to a process, indeed was known to regulate the genes associated with that process. For example, CLB2, a G2/M phase specific cyclin, had correlation 0.88 with process 12, which in turn has significant cell cycle related annotations. Even when information was not available to verify our proposed regulation relationships, the regulators were known to be related to glucose starvation, cell wall stress, cell growth, cyclic AMP, ribosome synthesis, nitrogen starvation, and mating, all processes known to be affected by the conditions in the Gasch *et al.* (2000) dataset.

5.2. Full COPR model

We next turned to the evaluation of our richer COPR model, which includes regulatory models for the different processes.

5.2.1. Statistical validation. We first tested whether the COPR model, which includes both the partition of genes into overlapping processes as well as the regulatory program of each process, resulted in improved statistical behavior compared to the simpler OP model, which did not include regulatory programs. To this end, we compared several models learned from the dataset of Gasch *et al.* (2001), specifying 20 processes, and using 1,000 genes whose expression levels varied in the dataset. For each model, we selected a different number of candidate regulators that the model could assign as parents of the activity levels of processes.

As a measure of performance, we first compared the likelihood that the different learned models assign to the expression data that was used to train the models, as this provides a measure of how well each model explains the input expression data. Specifically, we evaluated the probability $P(\mathbf{E}.Level \mid \mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}, \mathcal{M})$, where $\mathbf{G}.\mathbf{M}, \mathbf{A}.\mathbf{C}$ are the values of the hidden variables learned by the algorithm, and $\mathcal{M}$ is the learned model. As shown in Fig. 4, models that use a larger number of regulators assign a higher likelihood to the input expression data. This finding is quite surprising, as the expression level is independent of the regulatory model given the learned activity levels $\mathbf{A}.\mathbf{C}$. The model with no regulatory program can set these activity levels without constraints, attempting only to optimize this likelihood, whereas our COPR model is constrained to try to fit some regulatory program when selecting the same set of activity levels. In general, we would expect the unconstrained model to perform better, so the fact that the regulatory model provides higher likelihood indicates that the activity levels learned when we take into account regulatory models are actually more predictive of the expression levels. This behavior could result from an improved grouping of genes into processes, which would allow the process activity levels to more closely predict the actual gene expression levels.

However, a good fit to the training expression data does not necessarily imply that the model indeed captured true characteristics of the underlying domain. To test whether the COPR model captured meaningful properties, we evaluated the model performance on held out test data, using five-fold cross-validation over genes. Note that the values of the hidden variables—the gene process assignments—are unknown for

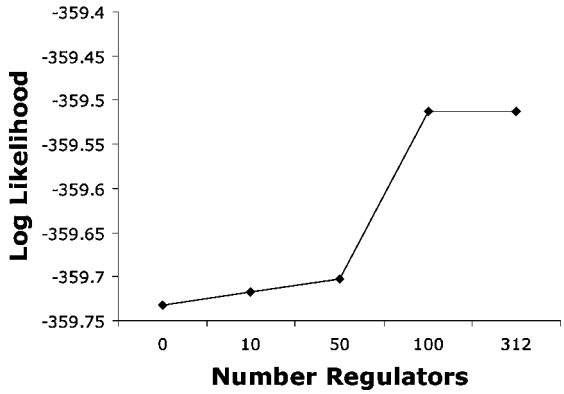

FIG. 4. Variation of training set likelihood as number of regulators available to the model is increased from zero.

genes in the test set. We evaluated the likelihood of the expression levels for the new genes by summing out over all possible assignments to the process memberships. Overall, the COPR model performed better than the OP model, assigning a higher likelihood to the held out test data in four out of the five cases, and achieving only a slightly lower likelihood in the fifth. The average likelihood per gene in the test set for the COPR model was -454.88, compared to -455.76 for the OP model A paired t-test on these results revealed a p-value of .04 for the improvement by the COPR model. Overall, these results indicate that the COPR model is, indeed, learning significantly better activity patterns than the model with no regulatory programs.

The experiment above demonstrates the ability of COPR to generalize to unseen genes. We also tested whether our COPR model can generalize to unseen experiments, by performing a similar five-fold cross validation scheme over arrays, using all 1,000 genes. In this case, the activity levels of the test set arrays are not known. Using the expression levels of regulators in the held out arrays, our regulatory model can predict the activity levels of each process in each of these new arrays and thus the expression levels for genes. We compared the COPR model's predictive power to a baseline model which uses no regulatory information and simply predicts expression level to be normally distributed with mean zero. (This very naive assumption is the best we can do with no predictive information regarding activity.) COPR outperformed the baseline model ($p = 0.02$), achieving an average log-likelihood per experiment of $-2{,}687.81$, as compared to $-3{,}097.42$ for the baseline model. These results indicate that regulatory programs learned by the COPR model are in fact predictive of the expression levels of their target genes.

5.2.2. Biological results. We next considered the ability of the COPR model and learning algorithm to reveal meaningful biological structure. We performed two experiments on different yeast datasets.

Yeast stress data. We first considered the same yeast stress expression data, learning a COPR model over 2,034 genes using 50 processes. Overall, the learned model had a reasonable partition of genes to processes, with 1,384 genes predicted to participate in exactly one process, 308 in two processes, 287 in three processes, and only 40 genes in four or more processes. The learned regulation programs were quite rich, including a total of 321 regulators out of the 466 candidate regulators from which the learning algorithm was allowed to select.

To analyze the biological plausibility of the assignments of genes to processes, as we did with the simple model, we tested whether the genes in each process were known to participate in the same process according to GO database of functional annotations (Ashburner, 2000). For each process p and each GO annotation t, we used the hypergeometric distribution to assign a p-value for the enrichment of genes in process p that are assigned to annotation t according to GO. Of the 50 learned processes, 33 were highly enriched for some known biological function, with $p < 0.001$. We compared the p-value of these enrichments to the p-value enrichments obtained using the OP model. Our results, summarized in Fig. 5(a), show that COPR model learned processes whose genes are significantly more enriched than those learned in the OP model, where 158 annotations were significantly more enriched in the COPR model.

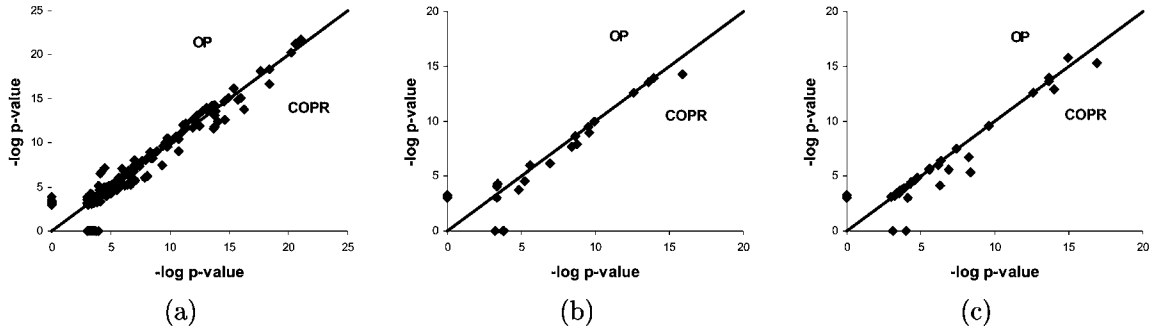

FIG. 5. Comparison of the simple OP and the COPR models on yeast stress data. Each graph shows a scatter plot of the negative log p-value for the enrichment of processes for different biological properties, comparing processes in the OP model (Y axis) and processes in the COPR model (X axis). (**a**) GO annotations; (**b**) presence of known motifs; (**c**) known transcription factor targets.

As the COPR model also learns the regulatory program for each process, we expect that genes assigned to the same process are also co-regulated. To test this hypothesis, we compiled a list of known binding sites from Wingender *et al.* (2001) and associated each gene with the binding sites that are contained within its 500 bp upstream region sequence. We then tested whether genes that were assigned by the COPR model to the process were enriched for any of these known binding sites, where again we used the hypergeometric distribution to assign a p-value to each such test. Overall, 19 of the 50 processes contained genes with some shared binding site with $p < 0.001$. We also examined process enrichment for targets of 106 transcription factors, using the genomewide protein–DNA binding data of Lee *et al.* (2002). We found that 20 of the 50 processes were enriched for targets of some transcription factor with $p < 0.001$.

A comparison to the OP model, shown in Fig. 5(b) and (c), shows that genes assigned to the same process according to the richer COPR model were significantly more enriched for motifs as well as transcription factor targets compared to genes that were assigned to the same process according to the OP model. Thus, we conclude that the inclusion of regulatory programs significantly improved our ability to detect truly co-regulated sets of genes.

A more in-depth examination of our results revealed several cases where our processes were a particularly good fit to current biological knowledge. In particular, eleven processes were enriched for a motif m and an annotation t, where the transcription factor known to bind motif m has a known regulatory role in regulating process t. As one example, process 48 contained 13 genes, 10 of which were known to participate in glycolysis ($p < 9 \cdot 10^{-22}$). In addition, 10 of its member genes shared the binding site for the transcription factor GCR1, which is known to control glycolysis. From the binding data of Lee *et al.* (2002), we also found that this process was enriched for targets of GCR2, which is also involved in glycolysis control. As another example, process 16 was enriched for protein folding and heat shock genes ($p < 1.8 \cdot 10^{-19}$). Importantly, genes in this process were also enriched for the binding site of the transcription factor HSF, a known regulator of protein folding and heat shock proteins. The binding data also indicated that this process was enriched for known targets of HSF, agreeing with both the GO and binding enrichments.

Next, we analyzed the quality of the actual regulatory programs that we learned for each process. In terms of their overall structure, the learned regression trees were plausible: most trees had one or two splits, no tree had more than four splits, and the majority of leaves had one linear parent. Our COPR model had multiple instances in which a regulator was used in more than one place in the regulation program. In most of these cases, the reused regulator appeared as a parent in the linear Gaussian leaf models of the regulatory program and not in the tree splits. This finding suggests that these two parts of the model correspond to different types of regulation and that linear regulators can play a role in different combinatorial contexts; both of these conclusions are consistent with biological knowledge.

We also attempted to validate the actual regulators assigned in each regulatory program. Such validation is, by necessity, somewhat anecdotal, as the biological knowledge regarding the role of the regulators is very limited. Nevertheless, our analysis found ten processes in which the regulator had a known function in regulating the GO function that was enriched in the genes of the process.

Finally, there were many cases in which our learned model suggested novel hypotheses regarding gene regulation. For example, one of the processes we learned contained 16 genes with an unknown function. However, in the motif enrichment analysis, we found seven motifs that were significantly enriched for the genes in this process (each occurring in at least 14 of the 16 genes). Moreover, two of the regulators assigned as part of the learned regulatory program, STE2 and GAC1, were associated with two of the enriched motifs. Combined with the coherent expression levels that we observed for these 16 genes, this result suggests a novel biological process and a putative regulatory program for controlling its activation.

Combined yeast datasets. As our COPR model allows genes to participate in multiple processes, we can apply it to large compendia of expression measurements that correspond to many different conditions, in order to study both the common and the specific regulatory relationships in these different conditions. To this end, we compiled a large compendium of 394 yeast microarrays from four different studies, including measurements of expression during the cell cycle (Spellman *et al.*, 1998), in response to various stress conditions (Gasch *et al.*, 2000, 2001), and in response to different gene deletion mutations (Hughes *et al.*, 2000).

From this combined dataset, we learned a COPR model over 5,747 genes, using 50 processes and 464 candidate regulators. To evaluate the partition of genes to processes, we performed the same enrichment

evaluation as above, testing the enrichment of genes assigned to the same process for GO annotations (Ashburner, 2000), cis-regulatory motifs (Wingender *et al.*, 2001), and for binding targets of 106 transcription factors (Lee *et al.*, 2002). The results are shown in Fig. 6. Overall, we found 36 processes that were enriched for at least one GO annotation with $p < .001$, 27 processes significantly enriched for DNA binding motifs, and 22 that were enriched for the targets of one of the 106 transcription factors assayed by Lee *et al.* In total, 45 of the 50 processes had some significant enrichment from one of the three sources.

We compared our results with those of the module network framework (Segal *et al.*, 2003b), with the hypothesis that a model allowing for overlapping processes would perform better on data compiled from different studies and spanning a wide range of experimental conditions. We note that unlike our COPR model, in the module networks framework, the regulators are also included in the gene modules, so the total number of genes grouped is slightly (about 10%) larger. As shown in Fig. 6, we found that our COPR model had noticeably more significant enrichments for GO annotations, regulatory motifs, and known transcription factor targets. In particular, out of 410 GO annotations enriched in either model, 312 were more significantly enriched in the COPR model than in the module networks model. Similarly, of the 32 motifs found to be enriched for either model, 28 were more enriched in the COPR model, and of the 52 transcription factor targets enriched in either model, 40 were more enriched in the COPR model. These results support the advantages of the two significant extensions of COPR over the module network model: allowing genes to participate in multiple processes and extending the regulatory model to encompass linear regulation.

In analyzing the learned regulatory programs, we found nine processes in which at least one of the predicted regulators had a known role in regulating the function associated with the GO annotation enriched for the process. As discussed, this type of analysis is limited by our very incomplete knowledge of regulatory relationships in the biological literature.

We next examined the predicted activity levels for each array. These are interesting as they provide evidence about cellular activity in different conditions; moreover, as these levels are predicted in part by the regulatory programs, they provide us with indirect evidence regarding the validity of these programs. We tested the correspondence between the activity levels and the actual conditions represented by each array. As we expect some processes to be differentially regulated in different conditions (e.g., heat shock proteins should be activated under various stress conditions), such correspondence would provide evidence that the learned activity levels correspond to true regulatory patterns.

Specifically, we compiled a list of 74 annotations for the different arrays, which specify the experimental condition that is represented by each array. For example, our list included annotations for the various cell cycle phases, as well as annotations for the different stress conditions to which the yeast was subjected (e.g., heat shock, nitrogen depletion). To measure how well our activity levels corresponded to these annotations, we performed a student t-test for each process p and each experiment annotation t, which measured whether the distribution of the activation levels was different between the arrays labeled with annotation t and those that are not labeled with it.

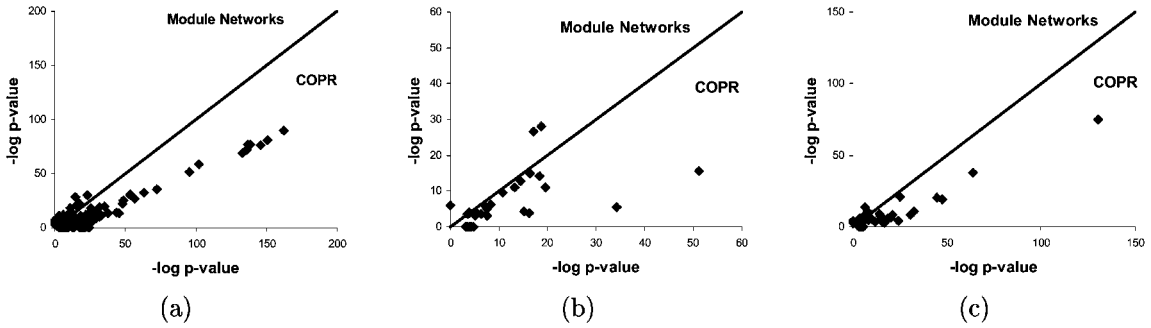

FIG. 6. Comparison of the module network model of Segal *et al.* to our COPR model on the yeast compendium data. Each graph shows a scatter plot of the negative log p-value for the enrichment of processes for different biological properties, comparing processes in the module network model (Y axis) and processes in the COPR model (X axis). (**a**) GO annotations; (**b**) presence of known motifs; (**c**) known transcription factor targets.

TABLE 1. A SAMPLE OF ACTIVITY LEVEL CORRELATIONS WITH EXPERIMENT ANNOTATIONS, PAIRED WITH RELEVANT GO ANNOTATIONS

Process	*Experiment annotation*	*t-test*	*GO annotation*	*p-value*
47	Mating	4.455	Mating	2.202e−9
46	Hyperosmolarity	12.083	Plasma membrane	2.65e−17
45	Protein modification	27.160	26S proteasome	2.187e−54
38	Mitochondrion	3.527	Mitochondrial inner membrane	3.11e−22
36	Heat	4.183	Heat shock protein	8.76e−05
26	Peroxisome organization	9.293	Peroxisomal matrix	5.94e−09
19	Stationary phase	3.651	S phase of mitotic cell cycle	1.91e−20
17	Chaperone activity	5.262	Chaperone	7.06e−20

For all 50 processes, we found that their activation levels had a high correspondence with at least one annotation with $p < 0.025$. In 19 cases, the experiment annotation corresponded to the significant GO annotation of the process. For example, the activation levels for the (GO annotated) protein folding process had a high correspondence to the stress annotation. Protein folding is known to be activated in response to various stress conditions, such as heat and exposure to certain chemicals. A partial list of the significant associations between activation levels and experiment annotations is shown in Table 1. We also compared the experiment annotations associated with a process to transcription factors associated with it via motif or transcription factor target enrichment. Overall, we found 11 processes where the experiment annotation matched the function of the associated transcription factor.

In many cases, the genes in the process, and the activity profile associated with the process, appear to define a very coherent biological unit. For example, process 43 was enriched for the targets of three mitosis regulating transcription factors from the genomewide protein–DNA binding data. This process was also enriched for a motif related to mitosis. The activity levels of this process were lowered significantly for the M-G1 phase annotated experiments (t-test value -3.18), and above normal for M-phase annotated experiments and experiments annotated with chromatin modification. This process has only a weak GO enrichment for the chromatin annotation, but the other evidence strongly suggests that it is associated with mitosis, perhaps with the transition from the M phase to the G1 phase in the cell cycle.

As another example, process 47 was strongly associated with GO annotations related to mating processes: "mating" ($p = 2.02x10^{-9}$), "pheromone response" ($p = 9.67x10^{-6}$), and zygote formation ($p = 1.5x10^{-5}$). Correspondingly, the activity level profile showed increased activity for experiments annotated for mating, pseudohyphal growth, and lowered activity in conditions such as salt stress, where we expect mating behavior to be repressed. The regulator RME1, which is associated with meiosis, was included in the learned program of this process. The process was strongly associated with two other transcription factors, MCM1, which regulates DNA replication, and STE12, which is associated with pheromones, exhibiting enrichment both for the targets of these two transcription factors in the protein–DNA binding data and for their motif in the genes' promoter regions. This process has strong support in the biological literature, which suggests that MCM1 and STE12 interact in the regulation of mating (Mead *et al.*, 2002) and even suggests a mating pathway in which MCM1 and our predicted regulator RME1 are both involved (Frenz *et al.*, 2001; Kunoh *et al.*, 2000b, 2000a).

6. CONCLUSIONS AND FUTURE WORK

This paper proposes a probabilistic framework for modeling overlapping cellular processes and extracting these models from gene expression data. We discussed a simple overlapping process model which includes only gene membership in processes and process activity, and an extended model—the COPR model—which includes regulatory programs for each process.

Our results demonstrate the advantage of allowing genes to participate in more than one process and validate the biological assumptions that underlie our probabilistic model. In the results for our richer COPR

model, we have shown that, by forcing our model to be consistent with regulatory programs, we obtain a better explanation of the data and processes that are biologically much more coherent.

We have also shown several cases where the predicted regulatory programs are consistent with current biological knowledge. In many cases, our COPR model's predictions are remarkably coherent: A process associated with a certain cellular function is often predicted to be active in precisely the conditions where that function plays a role. Such coherent results involving uncharacterized genes or regulators can suggest novel biological hypotheses that can be tested in the lab.

There are several possible extensions to our work. First, our use of a unified probabilistic framework easily allows the modular integration of additional data sources. For example, rather than using the protein–DNA binding data of Lee *et al.* (2002) solely for validating our result, we can directly integrate it into our model as a noisy sensor for regulation (as in Segal *et al.* [2002]). Another direction is the application of our framework to Affymetrix microarray technology, which measures the absolute expression level of a gene rather than its level relative to some control. Our approach, constrained to include only nonnegative activity levels, might provide a good decomposition of the observed expression levels into overlapping processes. Finally, it would be valuable to apply our framework to the analysis of cellular processes and regulation in human microarray data, where we expect the regulatory programs to be quite complex due to tissue-specific gene activation, so that genes are likely to play multiple roles.

ACKNOWLEDGMENTS

This work was supported by NSF grant ACI-0082554 under the ITR Program. Eran Segal was also supported by a Stanford Graduate Fellowship.

REFERENCES

Alter, O., Brown, P.O., and Botstein, D. 2000. Singular value decomposition for genome-wide expression data processing. *Proc. Natl. Acad. Sci.* 97(18), 10101–10106.

Ashburner, M. *et al.* 2000. Gene ontology: Tool for the unification of biology. *Nature Genet.* 25, 25–29.

Bjorck, A. 1996. *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA.

Breiman, L., Friedman, J.H., Olshen, R., and Stone, C. 1984. *Classification and Regression Trees*, Wardsworth International Group, Belmont, CA.

Cheng, Y., and Church, G.M. 2000. Biclustering of expression data. *ISMB'00*.

Dempster, A.P., Laird, N.M., and Rubin, D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. B* 39, 1–39.

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

Frenz, L., Johnson, A., and Johnston, L. 2001. Rme1, which controls CLN2 expression in *Saccharomyces cerevisiae*, is a nuclear protein that is cell cycle regulated. *Molecular Genetics and Genomics* 266(3), 374–384.

Friedman, N. 1998. The Bayesian structural EM algorithm. *Proc. UAI*.

Friedman, N., Nachman, I., and Peér, D. 1999. Learning of Bayesian network structure from massive datasets: The "sparse candidate" algorithm. Submitted.

Gasch, A.P., Huang, M., Metzner, S., Elledge, S.J., Botstein, D., and Brown, P.O. 2001. Genomic expression responses to DNA damaging agents and the regulatory role of the yeast ATR homolog meclp. *Mol. Biol. Cell* 12(10), 2987–3003.

Gasch, A.P., Spellman, P.T., Kao, C.M., Carmel-Harel, O., Eisen, M.B., Storz, G., Botstein, D., and Brown, P.O. 2000. Genomic expression program in the response of yeast cells to environmental changes. *Mol. Biol. Cell* 11, 4241–4257.

Geiger, D., and Heckerman, D. 2002. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *Ann. Statist.* 30(5), 1412–1440.

Hughes, T.R., Marton, M.J., Jones, A.R., Roberts, C.J., Stoughton, R., Armour, C.D., Bennett, H.A., Coffey, E., Dai, H., He, Y.D., Kidd, M.J., King, A.M., Meyer, M.R., Slade, D., Lum, P.Y., Stepaniants, S.B., Shoemaker, D.D., Gachotte, D., Chakraburtty, K., Simon, J., Bard, M., and Friend, S.H. 2000. Functional discovery via a compendium of expression profiles. *Cell* 102(1), 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.

Koller, D., and Pfeffer, A. 1998. Probabilistic frame-based systems. *Proc. AAAI*.

Kunoh, T., Kaneko, Y., and Harashima, S. 2000a. Positive regulation of transcription of homeoprotein-encoding YHP1 by the two-component regulator Sln1 in *Saccharomyces cerevisiae*. *Biochemical and Biophysical Research Communications* 278(2), 344–348.

Kunoh, T., Kaneko, Y., and Harashima, S. 2000b. YHP1 encodes a new homeoprotein that binds to the IME1 promoter in *Saccharomyces cerevisiae*. *Yeast* 16(5), 439–449.

Kyoto University Bioinformatics Center. 1995. KEGG: Kyoto encyclopedia of genes and genomes. *www.genome.ad.jp/kegg*.

Lazzeroni, L., and Owen, A. 1999. Plaid models for gene expression data. Technical report, Stanford.

Lee, T.I., *et al.* 2002. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science* 298, 824–827.

Lerner, U., and Parr, R. 2001. Inference in hybrid networks: Theoretical limits and practical algorithms. *Proc. 17th Ann. Conf. on Uncertainty in Artificial Intelligence*.

Mead, J., Bruning, A.R., Gill, M.K., Steinera, A.M., Acton, T.B., and Vershon, A.K. 2002. Interactions of the Mcm1 MADS box protein with cofactors that regulate mating in yeast. *Mol. Cell. Biol.* 22, 4607–4621.

Pe'er, D., Regev, A., Elidan, G., and Friedman, N. 2001. Inferring subnetworks from perturbed expression profiles. *ISMB '01*.

Segal, E., Barash, Y., Simon, I., Friedman, N., and Koller, D. 2002. From sequence to expression: A probabilistic framework. *Proc. RECOMB*.

Segal, E., Pe'er, D., Regev, A., Koller, D., and Friedman, N. 2003a. Learning module networks. *Proc. UAI*, Acapulco, Mexico, 525–534.

Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., Koller, D., and Friedman, N. 2003b. Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genet.*

Segal, E., Taskar, B., Gasch, A., Friedman, N., and Koller, D. 2001. Rich probabilistic models for gene expression. *Bioinformatics* 17(Suppl. 1), S243–S252.

Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., and Futcher, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell* 9(12), 3273–3297.

Tanay, A., Sharan, R., and Shamir, R. 2002. Discovering statistically significant biclusters in gene expression data. *Proc. ISMB. Bioinformatics* 18(Suppl. 1), S136–S144.

Wingender, E., *et al.* 2001. The TRANSFAC system on gene expression regulation. *Nucl. Acids Res.* 29, 281–283.

Address correspondence to:
Alexis Battle
353 Serra Mall
Computer Science Department
Stanford University
Stanford, CA 94305-9010

E-mail: ajbattle@stanfordalumni.org