Boosting The Performance Of Inference Algorithms For Transcriptional Regulatory Networks Using A Phylogenetic Approach

Xiuwei Zhang and Bernard M.E. Moret

Laboratory for Computational Biology and Bioinformatics
EPFL (Ecole Polytechnique Fédérale de Lausanne)
and Swiss Institute of Bioinformatics
Lausanne, Switzerland
{xiuwei.zhang,bernard.moret}@epfl.ch

Abstract. Inferring transcriptional regulatory networks from gene-expression data remains a challenging problem, in part because of the noisy nature of the data and the lack of strong network models. Time-series expression data have shown promise and recent work by Babu on the evolution of regulatory networks in *E. coli* and *S. cerevisiae* opened another avenue of investigation. In this paper we take the evolutionary approach one step further, by developing ML-based refinement algorithms that take advantage of established phylogenetic relationships among a group of related organisms and of a simple evolutionary model for regulatory networks to improve the inference of these networks for these organisms from expression data gathered under similar conditions.
We use simulations with different methods for generating gene-expression data, different phylogenies, and different evolutionary rates, and use different network inference algorithms, to study the performance of our algorithmic boosters. The results of simulations (including various tests to exclude confounding factors) demonstrate clear and significant improvements (in both specificity and sensitivity) on the performance of current inference algorithms. Thus gene-expression studies across a range of related organisms could yield significantly more accurate regulatory networks than single-organism studies.

1 INTRODUCTION

The widespread use in the life sciences of high-throughput gene-expression experiments has created a strong demand for suitable computational tools to analyze such data [19]. One of the most common analyses is the reconstruction of transcriptional (or gene) regulatory networks, models of the cellular regulatory system that governs transcription. In such networks, nodes are associated with genes or transcription factors, while arcs denote regulation. These networks can be inferred from gene-expression studies using various machine-learning or statistical inference methods. (Friedman [9] gave a survey of inference methods for probabilistic graphical models.) Other models have also been proposed, such as systems of differential equations [7].

Methods to reconstruct transcriptional regulatory networks from gene-expression data include Boolean networks and their generalizations [1, 15], Bayesian networks [10] and dynamic Bayesian networks (DBNs) [14], differential equations [7, 21], and many others. Results from these approaches, however, remain mixed: the high noise level in

the data, along with the lack of well defined, realistic models for the regulatory networks, combine with other factors (such as the typically large number of genes tested vs. the small number of test samples—the so-called "tall dataset" problem) to make inference difficult. Researchers recognized early that additional information about gene expression could be used to good effect. Methods were developed to use time-series expression data [5, 14, 22, 24]. Babu and his colleagues recently pioneered an evolutionary approach to the study of regulatory networks in *E. coli* and in *S. cerevisiae* [2–4, 20]. They posit a simple evolutionary model for regulatory networks, which adds or removes edges to the network, and proceed to investigate how well such a model accounts for the dynamic evolution of the two most studied regulatory networks. Using a similar evolutionary model, Bourque and Sankoff [6] developed an algorithm to improve the inference of cross-species gene networks based on their phylogenetic relationships.

We use a phylogenetic approach to develop algorithms that boost the performance of any chosen network inference method by using phylogenetic information and a simple model of network evolution. We consider a scenario where orthologous regulatory networks have been (separately) inferred for a number of related organisms whose phylogenetic relationships are known. Our algorithms refine these networks by considering all of them at once, within the known phylogeny of the organisms, to produce networks with much higher specificity and sensitivity. Whereas Bourque and Sankoff used a parsimony approach and tightly integrated inference and refinement, our algorithms are formulated within a maximum likelihood (ML) framework and focus solely on refinement, thereby allowing one to use one's preferred network inference method. We test our algorithms on various types of simulated data with different standard network inference approaches, and for various specificity and sensitivity settings, and compare the results with a standard approach. The *receiver-operator characteristic (ROC)* curves for our algorithms consistently dominate those of the standard approaches; under comparable conditions, they also dominate the results from Bourque and Sankoff. We also investigate the source of these improvements to eliminate various confounding factors.

This paper is organized as follows: we provide some background in Sec. 2, present our algorithms in Secs. 3 and 4, describe our experimental design in Sec. 5, and discuss our results in Sec. 6.

2 BACKGROUND

Our approach relies on placing inferred networks (obtained by one's favorite method) at the leaves of the known phylogenetic tree, reconstructing ancestral regulatory networks within this tree according to a model of network evolution, and propagating ancestral information back down to the leaves to improve the inferred networks. We use inference algorithms based on DBN and on differential equations. For the former, we use Murphy's Bayesian Network Toolbox [17]; for the latter, we use TRNinfer [21]. For ancestral sequence estimation, we use FastML [18].

2.1 DBNs for Network Inference

When DBNs are used to model regulatory networks, an associated structure learning algorithm is used to infer the networks from gene-expression data [8, 11, 14, 15]. The implementation of this algorithm in the Bayesian Network Toolbox provides two optimization functions: an ML score and a Bayesian information criterion (BIC) score.

Let D denote the dataset used in learning and G the (structure of the) network; the algorithm using ML scoring aims to return the structure $G^* = \arg\max_G \log Pr(D|G)$. However, transcriptional regulatory networks are typically sparse graphs, so that ML inferences often produce many false positive edges. The BIC score introduces a penalty on the complexity of G,

$$\log Pr(D|G, \hat{\Theta}_G) - 0.5 \#G \log N \tag{1}$$

where $\hat{\Theta}_G$ is the ML estimate of parameters for G, N is the number of samples in D, and $\#G$ is the number of free parameters of G. This penalty makes inference more conservative, reducing the number of false positives in the networks inferred by maximizing the BIC score. In practice, heuristic search methods are used, as well as mild restrictions on the structure of the model, the latter aimed at reducing the number of possible network structures—such as a small bound on the maximum indegree of the nodes, a restriction that appears well supported by the data [1, 15].

2.2 Differential Equations for Network Inference

Differential equations can describe causal relationships among components in a quantitative manner and are thus well suited to model transcriptional regulatory networks [7, 21]. A regulatory system is represented by the equation $dx/dt = f(x(t)) - Kx(t)$, where $x(t) = (x_1(t), \cdots, x_n(t))$ denotes the expression levels of the n genes and K (a matrix) denotes the degradation rates of the genes. The regulatory relationships among genes are then characterized by $f(\cdot)$. [21] produced a tool, TRNinfer, that solves the differential equations by formulating them into linear programming problems.

2.3 ML-Based Reconstruction of Ancestral Nodes

Reconstructing ancestral information in phylogenetic work is typically in the nature of an anchoring step in the computation, particularly in parsimony-based approaches. When we have high confidence in the tree and the edge lengths are modest, however, an ML approach to ancestral inference can yield accurate results; FastML [18], using a user-specified character substitution matrix, infers labels for the internal nodes (on a site-by-site basis) that maximize the overall likelihood of the tree. The algorithm was initially designed for protein sequences, but can be used for any type of sequences with a suitable substitution matrix.

Fix a site, i.e., a character position in the sequence. Let i denote a node in the tree, l_i the length of the edge between node i and its parent, and a the value of a character at some node, chosen from a given set S of possible character values. For each character a at each node i, we maintain two variables:

- $L_i(a)$: the likelihood of the best reconstruction of the subtree with root i given that the parent of i is assigned character a.
- $C_i(a)$: the optimal character assigned to i given that its parent is assigned as a.

Finally, let π_a denote the initial distribution of character a and $p_{ab}(l)$ the probability of substitution of a with b along an edge of length l. For simplicity, assume that the given tree is binary; then our adaptation of the FastML algorithm carries out these steps:

1. If leaf i has character b, then, for each $a \in S$, set $C_i(a) = b$ and $L_i(a) = p_{ab}(l_i)$.

2. If i is an internal node and not the root, its children are j and k, and it has not yet been processed, then, for each $a \in S$, set
   - $L_i(a) = \max_{c \in S} p_{ac}(l_i) \times L_j(c) \times L_k(c)$
   - $C_i(a) = \arg\max_{c \in S} p_{ac}(l_i) \times L_j(c) \times L_k(c)$
3. If there remain unvisited nonroot nodes, return to Step 2.
4. If i is the root node, with children j and k, assign it the value $a \in S$ that maximizes $\pi_a \times L_j(a) \times L_k(a)$.
5. Traverse the tree from the root, assigning to each node its character by $C_i(a)$.

3 APPROACH

The input is a set of gene-expression data matrices, for several related organisms, along with a known phylogeny (with edge lengths) for this group of organisms. (Such phylogenies are typically well established though the edge lengths remain to be explored.) The first step is simply to run one's preferred algorithm for regulatory network inference, independently on each of the data matrices; in this study, we use two types of inference algorithms, respectively based on DBN and differential equations. The resulting networks are used to label the corresponding leaves of the phylogeny. We encode a network by the concatenation of the rows of its adjacency matrix—every code thus represents a valid network. Note that the initial networks themselves are the real inputs to our algorithm; we use the gene-expression data stage in our tests solely in order to enhance the verisimilitude of our simulations.

We then use our adaptation of the FastML algorithm to infer ancestral networks, which in turn are used to refine the sequences at the leaves. We present below two algorithms to carry out this refinement, both based on the intuition (verified in simulations) that ancestral sequences are more accurate than those at the leaves, but only up to some height in the tree—as distant ancestral sequences suffer from the inference errors of FastML. The two middle steps can be iterated: starting from the newly refined networks, we can once again infer ancestral networks and use the results to refine the leaves.

We realize that edge lengths obtained from an analysis of the sequences of (typically) a few genes need not reflect the amount of evolution in the regulatory networks—while both evolved on the same tree, their respective rates of evolution could differ considerably. As we still lack the knowledge required to formulate a more precise model of network evolution, using the same edge lengths is just the neutral choice.

4 METHODS

4.1 Inferring the Initial Networks

In our study, we want to examine the ROC curves and so need to be able to trade off specificity and sensitivity. To this end, we modify the inference method based on DBN by generalizing Eq. 1 with a *penalty coefficient* k_p to adjust the penalty:

$$\log Pr(D|G, \hat{\Theta}_G) - k_p \#G \log N \tag{2}$$

where k_p varies from 0 to 0.5. With $k_p = 0$, we have the ML score; with $k_p = 0.5$, Eq. 2 reduces to Eq. 1. For the TRNinfer algorithm, the parameter that it provides to adjust the sparseness of the networks does not afford sufficient control to generate sparse enough

networks. We thus supplement it by applying different thresholds to the output connection matrix to choose final edges. We shall refer to these modified inference methods as *DBI* for that based on the DBN model and as *DEI* for that based on TRNinfer.

4.2 Inferring the Ancestral Sequences

In this study our adjacency matrices are binary, with a 1 in the (i, j) entry denoting an edge from node i to node j. We use binary matrices for simplicity's sake: generalization to weighted matrices is immediate and, indeed, the additional information present in a weighted matrix should further improve the results. The data used by FastML are thus:

- the proportions of 0s and 1s in the networks, $\Pi = \begin{pmatrix} \pi_0 & \pi_1 \end{pmatrix}$
- the topology of the phylogenetic tree;
- the *edge length* l_e of each edge e, i.e., the number of changes along this edge;
- for each edge length l_e, its corresponding substitution matrix, $P_s(l_e)$, which represents the mutation probability between 0 and 1.

The substitution matrices depend on edge length: the longer the edge, the higher the mutation probabilities. We choose a $P_s(1)$ for edge length 1 and calculate $P_s(l_e)$ for $l_e \geq 2$ using an exponential distribution, $P_s(l_e) = P_s^{l_e}(1)$.

4.3 Refining the Leaves

The underlying principle is simple: phylogenetically close organisms are likely to have similar regulatory networks; thus independent network inference errors at the leaves get corrected in the ancestral reconstruction process. Obviously, however, if too much evolution occurred, the ancestral reconstruction process itself generates errors. Thus a crucial aspect of our algorithm is how to use ancestral networks at various heights above the leaves to refine the leaves. We ran large series of experiments under various conditions (not shown); all showed an expected increase in accuracy when moving to the parents of the leaves, eventually replaced by a decrease when moving too far above the leaves. On the basis of our results, we chose to use only the immediate parents of the leaves for refinement—but note that these parents are themselves the product of a global ML inference and thus reflect the structure of the entire phylogeny.

4.4 A Fast Oblivious Refinement Algorithm

Our first algorithm, *RefineFast*, is designed to run quickly; it reposes complete trust in the networks associated with the parents of the leaves, using them to replace, rather than refine, the leaf networks.

1. From the current leaves, infer ancestral nodes using FastML.
2. For each leaf, pick its parent and evolve it (according to the length of the edge to the leaf and its substitution matrix) to generate a new child.
3. Use these new children to replace the old leaves.
4. Repeat Steps 1–3 until the total size of the leaf networks stabilizes.

We can use the same substitution matrices $P_s(l_e)$ in both Step 1 and Step 2, but choosing different substitution matrices can accelerate convergence.

The algorithm is deliberately oblivious: it uses the original networks only in the ancestral reconstruction, after which it replaces them with a sample network drawn from the distribution of possible children of the parent. When the original networks are noisy (a common occurrence), this simplistic procedure does quite well.

4.5 A Nonoblivious Refinement Algorithm

To use the information still present in the original leaf networks in the refinement step, we developed an ML-based refinement algorithm, *RefineML*. To use the existing leaf sequences, we assign each site of each leaf a belief (confidence) coefficient, k_b, which varies between 0.5 and 1. In the DBN framework, these coefficients can be calculated from the *conditional probability table* (CPT) parameters of the predicted networks. For each leaf i, we calculate the variables $L_i(a)$ and $C_i(a)$, as defined in Sec. 2.3, where a is the character value of the parent of leaf i inferred by FastML. Fix a site; then, using the notation introduced in Sec. 2.3, the *RefineML* algorithm can be described as follows:

1. Learn the CPT parameters for the leaf networks reconstructed by the base inference algorithm and calculate the belief coefficient k_b for every site.
2. From the current leaves, infer ancestral sequences using FastML.
3. For each leaf i with value b, let $Q_i(c) = k_b$ if $b = c$, $1 - k_b$ otherwise; then set
   - $L_i(a) = \max_{c \in S} p_{ac}(l_i) \times Q_i(c)$
   - $C_i(a) = \arg\max_{c \in S} p_{ac}(l_i) \times Q_i(c)$
4. For each leaf i, assign its most likely character from the variable $C_i(a)$.

5 EXPERIMENTAL DESIGN

In order to evaluate the accuracy gains provided by our boosting algorithms, we need simulated data, as they allow us to control the parameters and, more importantly, to get an absolute assessment of accuracy. Ideally, such simulation should be complemented by applications to real data, but such data currently exist only for tiny examples—indeed, we hope that our positive results will incite researchers to collect such data on a larger scale. Simulations may lack realism and may also introduce a systematic bias in the results. We cannot assess the severity of the first problem, but we take specific precautions against systematic bias, in both the simulations and the analysis.

We generate test data from three pieces of information: the phylogenetic tree, the network at the root, and the substitution matrix (which is in turn influenced by the evolutionary rate). We use a wide variety of phylogenetic trees from the literature (of modest sizes: between 20 and 60 taxa) and several choices of root networks, the latter variations on part of the yeast network from the KEGG database [13], as also used by Kim *et al.* [14]; we also explore a wide range of evolutionary rates. Our networks are of modest size, with 16 genes each—this selection makes the gene-expression tables less "tall" and thus, at least in principle, less prone to generate errors in reconstruction, thus presenting a more challenging case for a boosting algorithm.

We need quantitative relationships in the networks in order to generate gene-expression data from each network. Therefore, in the data generation process, we use adjacency matrices with signed weights. Weight values are assigned to the root network, yielding a weighted adjacency matrix $A = (a_{ij})$. We can obtain the adjacency matrix for its child, $A' = (a'_{ij})$, by mutating A according to the substitution matrix and repeating as we traverse down the tree to obtain weighted adjacency matrices at the leaves. In other words, we evolve the weighted networks down the tree according to the model parameters—standard practice in the study of phylogenetic reconstruction [12, 16].

To generate gene-expression data from the weighted networks, we use both Yu's GeneSim [23] and *DBNSim*, our own design based on the DBN model.

5.1 Gene-Expression Data Generated by DBNSim

For *DBNSim*, we follow [15], using binary gene-expression levels, where 1 and 0 indicate that the gene is, respectively, *on* and *off*. Denote the expression level of gene g_i by x_i, $x_i \in \{0,1\}$; if m_i nodes have arcs directed to g_i in the network, let the expression levels of these nodes be denoted by the vector $y = y_1 y_2 \cdots y_{m_i}$ and the weights of their arcs by the vector $w = w_1 w_2 \cdots w_{m_i}$. From y and w, we can get the conditional probability $Pr(x_i|y)$. Once we have the full parameters of the leaf networks, we generate simulated time-series gene-expression data. At the initial time point, the expression level of gene g_i is generated by the initial distribution $Pr(x_i)$; at time t, its expression level is generated based on y at time $t-1$ and the conditional probability $Pr(x_i|y)$.

5.2 Gene-Expression Data Generated by GeneSim

GeneSim [23] can produce simulated gene-expression values for a given weighted network as well as generate arbitrary network structures. In contrast to our *DBNSim* method, GeneSim gives continuous gene-expression levels. Denoting the gene-expression levels of the genes at time t by the vector $x(t)$, the values at time $t+1$ are calculated according to $x(t+1) = x(t) + (x(t) - z)C + \varepsilon$, where C is the weighted adjacency matrix of the network, the vector z represents *constitutive expression values* for each gene, and ε models noise in the data. The values of $x(0)$ and $x_i(t)$ for those genes without parents are chosen uniformly at random from the range $[0, 100]$, while the values of z are all set to 50. The term $(x(t) - z)C$ represents the effect of the regulators on the genes; this term needs to be amplified for the use of *DBI*, because of the required discretization. We use a factor k_e with the regulation term (set to 7 in our experiments), yielding the new equation $x(t+1) = x(t) + k_e(x(t) - z)C + \varepsilon$.

5.3 Testing under Various Methods

With two data generation methods, *DBNSim* and GeneSim, and two network inference algorithms as our base algorithms, *DBI* and *DEI*, we can conduct experiments with different combinations of data generation methods and inference algorithms to verify that our boosting algorithms work under all circumstances. First, we use different data generation methods with the same inference algorithm. Since the binary gene-expression data generated by *DBNSim* does not fit *DEI*, we use *DBNSim* and GeneSim to generate data for *DBI*. We generate 200 time points for each gene-expression matrix, running the generation process 10 times to obtain mean and standard deviation. Second, we apply *DBI* and *DEI* to datasets generated by GeneSim to infer the networks. Since *DEI* does not accept large datasets (with many time points), here we used smaller datasets than the previous group of experiments with 75 time points, yielding expression level matrix of size 16×75. We run the generation process 20 times for each choice of tree structure and parameters and calculate mean and standard deviation. Finally, we conduct experiments with various evolutionary rates.

5.4 Comparing with the Bourque and Sankoff Approach

Bourque and Sankoff's algorithm [6], thereafter the B&S algorithm, also uses phylogenetic information to improve the inference of gene networks. We therefore conduct experiments, using continuous data, to compare our approach to theirs.

5.5 Where is the Important Information?

Although we use only the direct parents to refine the leaves at each iteration, the leaves receive information from the whole tree, since the FastML algorithm assigns states to every internal node based on global information. We claim that the use of this global information is necessary. To verify this claim, we build a variation of our algorithms, that we call *RefineLocal*, where the ancestral reconstruction stops once the parents of leaves are reached. The resulting ancestral reconstruction, in other words, is now limited to exactly the parts of the tree used in the leaf refinement. *RefineLocal* works with both *RefineFast* and with *RefineML*, since it does not alter the refinement phase of the algorithm.

Part of the improvement is due to noise averaging, taking advantage of the independence in errors among the leaf networks. We claim that noise averaging not based on the correct phylogeny cannot produce the type of improvement we see. To verify this claim, we build a procedure that we call *RefineRandomTree*, which runs our full refinement procedures (either one), but does it on a tree where the initial inferred networks were randomly assigned to leaves. Since the tree topology is unchanged, the averaging effect over the data remains globally similar, but the phylogenetic relationships are destroyed. We run 100 such randomized tests and report the mean behavior.

5.6 Measurements

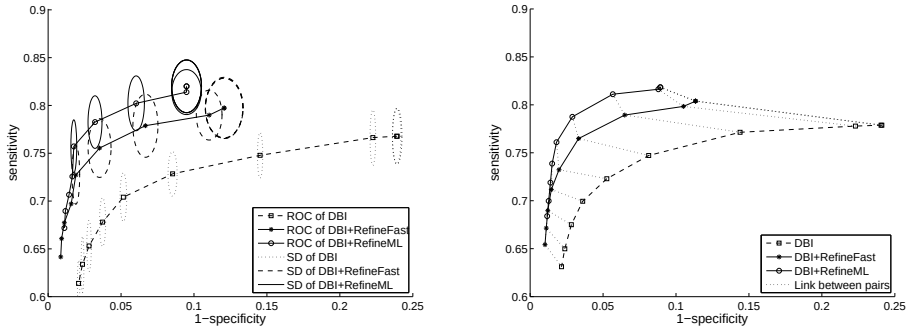
We want to examine the predicted networks at different levels of sensitivity and specificity. For *DBI*, on each dataset, we apply different penalty coefficients to predict regulatory networks, from 0 to 0.5, with an interval of 0.05, which results in 11 discrete penalty coefficients. For each penalty coefficient, we apply *RefineFast*, *RefineML*, *RefineLocal*, and *RefineRandomTree* on the predicted networks. For *DEI*, we also choose 11 thresholds for each predicted weighted connection matrix to get networks on various sparseness levels. For each threshold, we apply *RefineFast*, *RefineLocal*, and *RefineRandomTree* on the predicted networks. We measure specificity and sensitivity to evaluate the performance of the algorithms and plot the values, as measured on the results for various penalty coefficients (for *DBI*) and thresholds (for *DEI*) to yield ROC curves. Recall that in such plots, the larger the area under the curve, the better the results.

6 RESULTS AND ANALYSIS

Space constraints prevent us from showing more than a sample of our results. We show results on two representative trees: tree *T1* has 41 nodes on 6 levels and is better balanced than tree *T2*, which has 37 nodes on 7 levels. Both trees were generated with an expected evolutionary rate of 2.2 events (gain or loss of a regulatory arc in the network) per edge and resulting leaf networks have from 23 to 38 edges.

6.1 On Boosting under Different Experimental Settings

Different gene-expression data generation methods, same inference algorithm: Fig. 1 shows the average performance of *RefineFast*, *RefineML*, and *DBI* on 10 noiseless datasets generated by *DBNSim* on trees *T1* (left) and *T2* (right). Throughout the range of parameters, our two algorithms clearly dominate *DBI*, with *RefineML* also dominating the simpler *RefineFast*: for every penalty coefficient, both sensitivity and specificity are improved from *DBI* to *RefineFast* and further improved from *RefineFast* to *RefineML*, as easily seen on the right. Sample standard deviations of sensitivity and specificity



(a) *Results on* T1*, with standard deviations of sensitivity and specificity*

(b) *Results on* T2*, with dotted lines indicating matched penalty coefficients*

Fig. 1. ROC curves for DBI and boosting algorithms on the datasets generated by DBNSim

for these three methods on the noiseless datasets on *T1* are shown as ellipses, the loci of one standard deviation around each point. The separation between the curves is almost always larger than the standard deviations, so that our assertions of dominance of one method over another hold, not only on average, but also in the vast majority of cases. Also, as this figure demonstrates, the boosting effect remains similar on different phylogenies—and so we present results only on *T1* hereafter.

All three algorithms behave on the noisy datasets much as on the noiseless ones. Our refinement algorithms yield more improvement on the noisy datasets, which are closer to the real data and thus cause more difficulties for *DBI* methods, yielding a larger margin for improvement. We thus show results for noiseless datasets only, as the level of improvement caused by our algorithms can only increase as the noise level in the data increases. Fig. 2 shows the results of the three algorithms on the noiseless datasets generated by GeneSim on *T1*. The boosting effects are much the same as seen in Fig. 1, but it is clear that the *DBI* base algorithm does worse on the datasets generated by GeneSim than on those generated by *DBNSim*, as might be expected.

Different inference algorithms, same gene-expression data generation method: The datasets used in this experiment are generated by GeneSim. Fig. 3 shows the ROC curves of *DBI* and *DEI*, along with *RefineFast* boosting, on the same datasets; the refinement algorithm clearly dominates the base algorithms.

Different evolutionary rates: The expected evolutionary rate (average edge length) was fixed in all experiments presented above. High rates of evolution cause various difficulties in phylogenetic reconstruction; we thus expect our method to become less effective as evolutionary rates increase. To study this problem, we conducted experiments on tree *T1* with a root network of 16 nodes and 24 edges, using different evolutionary rates to generate the leaf networks. Fig. 4 shows ROC curves for *RefineML* and *DBI* with evolutionary rates of 2.32, 4.76 and 6.67 on noiseless datasets. The loss in performance as the rate of evolution increases is clear for both methods; since *DBI* itself suffers (perhaps because some networks produced in the simulation violate implicit assumptions), the loss in performance of *RefineML* is a combination of worsened

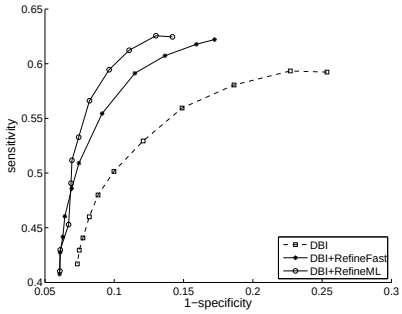


Fig. 2. ROC curves for DBI and boosting algorithms on the datasets generated by GeneSim

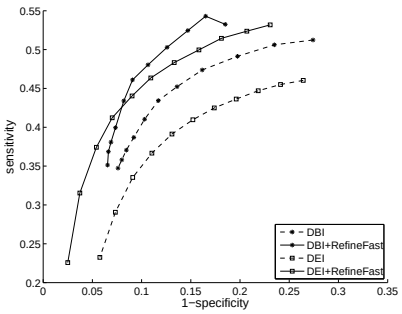


Fig. 3. Results for DBI and DEI with RefineFast boosting on GeneSim generated datasets

leaf networks returned by *DBI* and worsened ancestral reconstruction by FastML. Yet boosting is evident in all cases and performance remains excellent at the very high evolutionary rate of 4.76: most paths from the root to a leaf in the tree have 5 edges and so, at that rate, have an expected length of 23.5, so that the expected number of changes from the root network almost equals the number of edges of that network—a very challenging problem and one that is remarkably well solved here.

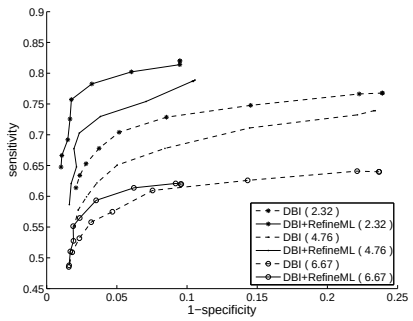


Fig. 4. ROC curves for DBI and RefineML under various evolutionary rates

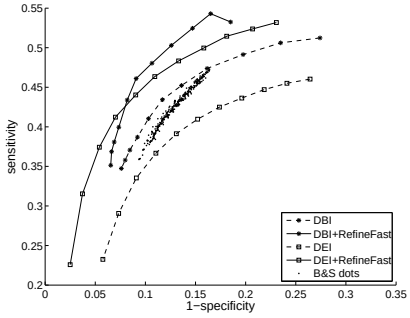


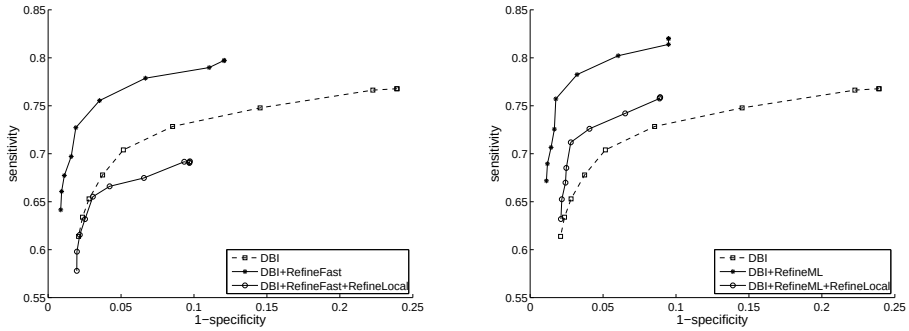
Fig. 5. Performance for B&S and RefineFast based on DBI and DEI

6.2 On Performance with respect to the B&S Algorithm

Since B&S requires continuous time-series gene-expression data, we use the same datasets, generated by GeneSim, as in Fig. 3. Fig. 5 presents the performance of B&S and *RefineFast* based on both *DBI* and *DEI*. The results of B&S are shown as a cloud of points, obtained under different parameter settings. B&S does better than plain *DEI*, but is clearly dominated by our *RefineFast* based on *DEI*, meaning that our refinement algorithm gains more improvement than B&S does.

6.3 On Applying ML Globally

We described earlier *RefineLocal*, a variant of our algorithms that infers ancestral networks only for the part of the tree that is used in the refinement phase. We use this algorithm to show that the improvement wrought in the leaves by our algorithms uses the phylogenetic information of the whole tree, not just the information present in the subforest induced by direct parents of leaves. Fig. 6(a) compares the performance of *RefineFast* with that of its localized version on noiseless datasets generated by *DBN-Sim* (the same datasets as in Sec. 6.1), while Fig. 6(b) does the same for *RefineML* on the same datasets. The plots are very similar: *RefineLocal* is clearly worse than the original algorithms, especially in terms of sensitivity. In fact, *RefineLocal* based on *RefineFast* does worse than *DBI*—due to the fact that the ancestral inference procedure introduces significant additional errors when limited to small subtrees. On the other hand, *RefineLocal* based on *RefineML* outperforms *DBI*—indicating that there is significant information present in the leaves, independent of the ancestral reconstruction.

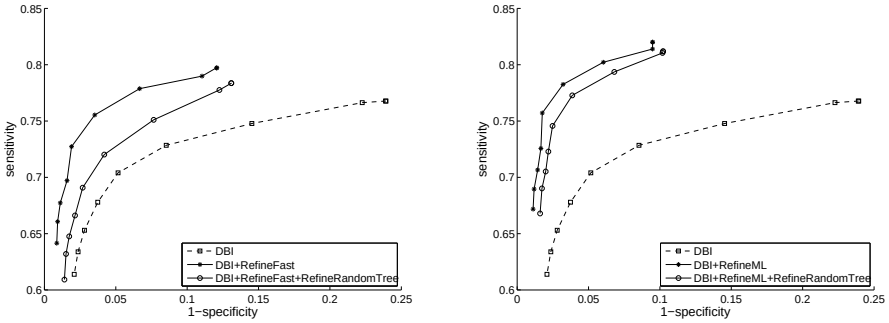


(a) *ROC curves with RefineFast* (b) *ROC curves with RefineML*

Fig. 6. ROC curves for DBI and RefineLocal, showing RefineFast (left) and RefineML (right)

6.4 On Phylogenetic Information

In Sec. 5.5 we introduced *RefineRandomTree*, which carries out our full algorithms, but on a tree where the leaves have been reshuffled randomly. Its purpose is to demonstrate that the improvements we observe are not due entirely to noise averaging among the leaf networks. Fig. 7 compares the performance of *RefineFast* (left) and *RefineML* (right) run on the correct phylogenetic tree with the average performance (over 100 runs) of the same algorithm run after randomly reshuffling leaf labels. Both *RefineFast* and *RefineML* show clearly worse performance on the reshuffled trees than on the correct one. The results on the shuffled trees are still better than the base algorithm *DBI*, which shows the error averaging effect of the trees. However, this improvement depends on the performance of the base algorithm: in other experiments (not shown) with larger gene-expression datasets, where *DBI* does better, *RefineFast* on the shuffled trees does not outperform *DBI*, while *RefineML* with shuffled trees does. Overall, the results



(a) *ROC curves with RefineFast* (b) *ROC curves with RefineML*

Fig. 7. ROC curves for DBI and RefineRandomTree, with RefineFast (left) and RefineML (right)

demonstrate the value of correct phylogenetic data, the value of the information present in the original leaf networks, and the averaging effect of the trees.

7 CONCLUSIONS AND FUTURE WORK

We described algorithms for boosting the accuracy of regulatory network inference on related organisms using a phylogenetic approach and gave experimental results demonstrating the effectiveness of these algorithms. Our algorithms yield significant gains in both sensitivity and specificity under all conditions and can, in principle, be used with any favorite network inference tool and any favorite phylogenetic reconstruction algorithm. Further assessments of the sensitivity of our methods to the accuracy of the branch length estimates and of their ability to correct for systematic errors in gene expression levels are planned.

Our approach requires comparable gene-expression data for orthologous regulatory networks, or the networks themselves, for a fair number of organisms; while cost and technical feasibility are no longer major obstacles to the accumulation of such data, ensuring comparability remains challenging, although we hope that our promising results will encourage researchers to consider collecting such data.

Many improvements to our approach are clearly possible, from refined models of network evolution to more precise handling of ancestral networks. Our approach to the use of phylogenetic information in network inference is not limited to transcriptional regulatory networks: it can be used, with suitable adaptations, for other types of signalling and metabolic networks.

REFERENCES

1. T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In *Proc. 4th Pacific Symp. on Biocomputing (PSB'99)*, pp. 17–28. World Scientific, 1999.
2. M.M. Babu, N.M. Luscombe, L. Aravind, M. Gerstein, and S.A. Teichmann. Structure and evolution of transcriptional regulatory networks. *Curr. Opinion in Struct. Bio.*, 14(3):283–291, 2004.

3. M.M. Babu and S.A. Teichmann. Evolution of transcription factors and the gene regulatory network in Escherichia coli. *Nucleic Acids Res.*, 31(4):1234–1244, 2003.
4. M.M. Babu, S.A. Teichmann, and L. Aravind. Evolutionary dynamics of prokaryotic transcriptional regulatory networks. *J. Mol. Bio.*, 358(2):614–633, 2006.
5. Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.
6. G. Bourque and D. Sankoff. Improving gene network inference by comparing expression time-series across species, developmental stages or tissues. *J. Bioinform. Comput. Biol*, 2(4):765–783, 2004.
7. T. Chen, H.L. He, and G.M. Church. Modeling gene expression with differential equations. In *Proc. 4th Pacific Symp. on Biocomputing (PSB'99)*, pp. 29–40. World Scientific, 1999.
8. R.C. Conant. Extended dependency analysis of large systems. *Int'l J. General Systems*, 14(2):97–141, 1988.
9. N. Friedman. Inferring cellular networks using probabilistic graph models. *Science*, 303(5659):799–805, 2004.
10. N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *J. Comput. Bio.*, 7(3-4):601–620, 2000.
11. N. Friedman, K.P. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proc. 14th Conf. on Uncertainty in Art. Intell. UAI'98*, pp. 139–147, 1998.
12. D.M. Hillis. Approaches for assessing phylogenetic accuracy. *Syst. Bio.*, 44:3–16, 1995.
13. M. Kanehisa, S. Goto, M. Hattori, K.F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.*, 34:D354–D357, 2006.
14. S.Y. Kim, S. Imoto, and S. Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, 4(3):228–235, 2003.
15. S. Liang, S. Fuhrman, and R. Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In *Proc. 3rd Pacific Symp. on Biocomputing (PSB'98)*, pp. 18–29. World Scientific, 1998.
16. B.M.E. Moret and T. Warnow. Reconstructing optimal phylogenetic trees: A challenge in experimental algorithmics. In R. Fleischer, B.M.E. Moret, and E.M. Schmidt, editors, *Experimental Algorithmics*, vol. 2547 of *Lecture Notes in Computer Science*, pp. 163–180. Springer Verlag, 2002.
17. K.P. Murphy. The Bayes net toolbox for MATLAB. *Computing Sci. and Statistics*, 33:331–351, 2001.
18. T. Pupko, I. Pe'er, R. Shamir, and D. Graur. A fast algorithm for joint reconstruction of ancestral amino acid sequences. *Mol. Bio. Evol.*, 17(6):890–896, 2000.
19. D.K. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature Genetics*, 32:502–508, 2002.
20. S.A. Teichmann and M.M. Babu. Gene regulatory network growth by duplication. *Nature Genetics*, 36(5):492–496, 2004.
21. R. Wang, Y. Wang, X. Zhang, and L. Chen. Inferring transcriptional regulatory networks from high-throughput data. *Bioinformatics*, 23(22):3056–3064, 2007.
22. R. Xu, X. Hu, and D.C. Wunsch. Inference of genetic regulatory networks from time series gene expression data. In *Proc. IEEE Int'l Joint Conf. on Neural Networks*, vol. 2, pp. 1215–1220. IEEE Press, Piscataway, NJ, 2004.
23. J. Yu, V.A. Smith, P.P. Wang, A.J. Hartemink, and E.D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.
24. W. Zhao, E. Serpedin, and E.R. Dougherty. Inferring gene regulatory networks from time series data using the minimum length description principle. *Bioinformatics*, 22(17):2129–2135, 2006.