Using Phylogenetic Relationships to Improve the Inference of Transcriptional Regulatory Networks

Xiuwei Zhang Maryam Zaheri
Bernard M.E. Moret

Laboratory for Computational Biology and Bioinformatics, Swiss Federal Institute of Technology
EPFL INS LCBB, INJ 230, Station 14, CH-1015 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. We conjecture that established phylogenetic relationships among a group of related organisms can be used to improve the inference of regulatory networks for these organisms from expression data gathered under similar conditions. We develop an inference algorithm to take advantage of such information and present the results of simulations (including various tests to exclude confounding factors) that clearly show the added value of the phylogenetic information. Our algorithm and results offer support for our conjecture and indicate that gene-expression studies under identical conditions across a range of related organisms could yield significantly more accurate regulatory networks than single-organism studies.*

1. Introduction

The widespread adoption of high-throughput instruments (such as microarrays) for gene expression data has given rise to the development of many tools for the analysis of such data [20]. One of the most common analyses is the reconstruction of transcriptional (or gene) regulatory networks, models of the cellular regulatory system that govern transcription. Transcriptional regulatory networks describe in graphical form the interactions between genes and their associated transcription factors; they can be inferred from the result of gene expression studies using various machine-learning or statistical inference methods (for a survey of inference methods for probabilistic graphical models, see [9]). Nodes in such networks are associated with genes or transcription factors, while arcs denote activation (or inhibition, in some cases). Other models have also been proposed, such as systems of differential equations [6].

Many methods have been proposed to reconstruct transcriptional regulatory networks from gene-expression data, including Boolean networks and their generalizations [1, 15], Bayesian networks [10] and dynamic Bayesian networks (DBNs) [14, 18], 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 relatively 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 produce better results. For example, methods were developed to use time-series expression data [5, 14, 22, 23]. Over the last five years, Babu and his colleagues pioneered an evolutionary approach to the study of regulatory networks in *E. coli* and in *S. cerevisiae* [2, 3, 4, 21]. They posit a simple evolutionary model for regulatory networks, which amounts to adding edges to, or removing edges from, the network and proceed to investigate how well such a model accounts for the dynamic evolution of two of the best studied networks.

In this paper, we advance and test the hypothesis that an evolutionary approach can significantly improve the inference of regulatory networks for related organisms. Specifically, we consider a scenario where the same bench experiments (e.g., the same microarrays) have been run on a number of related organisms whose phylogenetic relationships are known and we assume that a standard approach has been used to infer a regulatory network for each of the organisms independently. We present an algorithm that improves

these networks by making use of the known phylogenetic relationships—in other words, our algorithm considers the various networks together rather than individually and is thereby able to produce networks with both higher specificity and higher sensitivity. We test our algorithm on various types of simulated data and for various specificity and sensitivity settings and compare the results with a standard approach; we find that the *receiver-operator characteristic (ROC)* curves for our algorithm consistently dominate those of the standard approach. Finally, we investigate the source of these improvements, eliminating various simple possibilities such as noise averaging and demonstrating that it is indeed the phylogenetic data that enables our algorithm to improve upon the standard approach.

The rest of this paper is organized as follows: we provide the necessary background in Section 2, present our algorithm in Section 3, describe our experimental design in Section 4, and discuss our results in Section 5, before concluding in Section 6 with suggestions for future work.

2. Background

Transcriptional regulatory networks are often modelled as directed graphs [8], with nodes representing the genes and arcs (directed edges) representing the regulatory relationships between these genes. Thus inference methods take as input an array of gene expression data and produce a directed graph with one node for each gene in the input data.

Our approach to the use of phylogenetic information in reconstructing regulatory networks relies upon placing inferred networks (inferred separately by a method of one's choice) at the leaves of the known phylogenetic tree, reconstructing ancestral regulatory networks within this tree, and propagating ancestral information back down to the leaves to improve the inferred networks. We thus need both a network inference method and a phylogenetic method for reconstructing data at ancestral nodes. We decided to use DBNs, which have been widely used for this purpose and also because we wanted to use probabilistic models in both network inference and ancestral sequence estimation. For implementation, we use Murphy's Bayesian Network Toolbox [17], which supports a wide array of probabilistic graphical models. For ancestral network estimation, we use a maximum likelihood (ML) approach; since ML algorithms for phylogenetic tree reconstruction typically do not attempt to reconstruct ancestral sequences, most standard ML packages could not be used, so we chose FastML [19].

2.1. DBNs for regulatory network inference

DBNs have been proposed to model regulatory networks and its structure learning algorithm is used to predict regulatory networks from gene-expression data [14, 18]. The structure learning algorithm has been described in various contexts [7, 11, 15]. Its implementation in the Bayesian Network Toolbox provides two scores to evaluate a network: an ML score and a Bayesian information criterion (BIC) score, both frequently used in the literature.

Let D denote the dataset used in learning and G the (structure of the) network; then the structure learning algorithm using ML scoring aims to return the structure $G^* = \arg\max_G \log Pr(D|G)$. Transcriptional regulatory networks are typically sparse graphs, so that ML inferences tend to produce a significant number of false positive edges. The BIC score introduces a penalty on the complexity of the structure to get a tradeoff between fit and complexity, which is defined as

$$\log Pr(D|G, \hat{\Theta}_G) - \#G\frac{\log N}{2} \quad (1)$$

where $\hat{\Theta}_G$ is the ML estimate of network parameters for structure G, N is the number of samples in dataset D, and $\#G$ is the dimension of structure G, defined as the number of free parameters of G. The penalty for model complexity makes networks inferred under this criterion more conservative, gaining specificity at the expense of sensitivity.

In practice, heuristic search methods are used, as well as mild restrictions on the structure of the model, the latter aimed at reducing the huge 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] and that we use in our simulations.

2.2. ML-based reconstruction of ancestors

While reconstructed ancestral information is not viewed as an attempt to characterize an ancestral lineage, it is nevertheless useful in systematics and evolutionary biology; it is also an essential feature of phylogenetic reconstruction based on maximum parsimony. If the phylogenetic tree is known, along with its edge lengths, then FastML [19], using a user-specified character substitution matrix, infers labels for the internal nodes that maximize the overall likelihood of the tree. The algorithm uses dynamic programming; it was initially designed for protein sequences, but can be used for any type of sequences with a suitable substitution matrix. Ancestral sequences are inferred site by site, as characters are assumed to be independently distributed.

3. Methods

3.1. Overview

The input is a set of gene-expression data matrices, collected under similar experimental conditions, for several related organisms, along with a known phylogeny (with edge lengths) for this group of organisms. Thus there are three dimensions to the data: the number of organisms (the number of matrices), the number of genes (the number of

rows in each matrix), and the number of test conditions (the number of columns in each matrix).

Our algorithm is, in effect, a booster; therefore, the first step is simply to run one's preferred algorithm for regulatory network inference, independently on each of the data matrices; the resulting networks are used to label the corresponding leaves of the phylogeny. Since we need sequences of characters as labels, we represent a network by the concatenation of the rows of its adjacency matrix.

Once this initialization is complete, our algorithm uses an adaptation of the FastML algorithm to infer ancestral sequences so as to maximize the resulting likelihood. It is crucial to use a sequence encoding of the networks (such as ours) in which every possible sequence corresponds to a legal network: in the ancestral inference step, sequences are inferred site by site, which precludes the imposition of formatting constraints.

The ancestral sequences are then used to refine the sequences at the leaves. Our algorithm to carry out this refinement uses an ML approach and is based on the intuition (verified in experiments) that ancestral sequences should show more accuracy than those at the leaves, but only up to some height in the tree—distant ancestral sequences would suffer from their own inference errors.

We are well aware of the fact that the edge lengths obtained for the phylogenetic tree from an analysis of the sequences of a few genes (the most common source of phylogenetic trees) need not reflect the amount of evolution in the regulatory networks. However, as discussed earlier, we currently lack the knowledge required to formulate a more precise model of network evolution; choosing to use the same edge lengths is thus just the neutral choice.

3.2. Inferring the initial networks

The inference algorithm we use to initialize the process is the *DBN*, as implemented in the Bayesian Network Toolbox [17]. In our application, however, we want to examine the ROC curves and so need to be able to trade off specificity and sensitivity. To this end, we generalize Eq. 1 and use a *penalty coefficient* k_p to adjust the extent of the penalty, obtaining the new function

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

where k_p varies from 0 to 0.5. With $k_p = 0$, we have the ML score; this is equivalent to the objective function used in REVEAL [15, 18], which maximizes the mutual information between parents and child. With $k_p = 0.5$, the score of Eq. 2 becomes the original BIC score from Eq. 1.

3.3. Inferring the ancestral sequences

We use 0-1 regulatory networks 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 and a 0 denoting the absence of an edge. 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 topology of the phylogenetic tree;
- the *edge length* l_e for each edge e, meaning the number of changes happening on this edge;
- the proportions of 0s and 1s in the networks

$$\Pi = \begin{pmatrix} \pi_0 & \pi_1 \end{pmatrix} \tag{3}$$

- for each edge length l_e, its corresponding substitution matrix—the mutation probabilities between 0 and 1

$$P_s(l_e) = \begin{pmatrix} p_{00}(l_e) & p_{01}(l_e) \\ p_{10}(l_e) & p_{11}(l_e) \end{pmatrix} \tag{4}$$

The substitution matrices differ for different edge lengths: the larger the edge length, the higher the mutation probabilities. We choose a $P_s(1)$ for edge length 1 and we calculate $P_s(l_e)$ for $l_e \geq 2$ according to an exponential distribution:

$$P_s(l_e) = P_s^{l_e}(1) \tag{5}$$

3.4. Refining the leaves

Once we obtain ancestral networks, we are ready to refine the networks at the leaves. The principle is clear: closely related (in a phylogenetic sense) organisms are likely to have similar regulatory networks and thus network inference errors at the leaves get corrected in the ancestral reconstruction process. Obviously, however, if too much evolution stands in the way of accuracy, the principle no longer holds. Thus a crucial aspect of our algorithm is how to use ancestral networks at various heights above the leaves to refine the leaves. To ascertain how the accuracy of ancestral reconstructions varies as a function of height, we ran large series of experiments under various conditions (not shown); these 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 decided to use only the direct parents of the leaves for refinement. (More distant ancestors still participate in the refinement, however, as the parents of the leaves are estimated by FastML in a global setting and thus depend on the sequences chosen for all of their ancestors.)

3.5. The refinement algorithm RefineML

RefineML refines existing leaf networks using the reconstructed networks of their immediate parents. In order 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. Now, for each leaf i, we calculate the storage variables $L_i(a)$ and $C_i(a)$, where a is the character value of the parent of leaf i inferred by FastML, and $L_i(a)$ and $C_i(a)$ can be explained as:

- $L_i(a)$: the likelihood of the best reconstruction of leaf i on the condition that its parent is assigned as a.
- $C_i(a)$: the optimal character assigned to i on the condition that its parent is assigned character a.

These variables and the direct parents inferred by FastML give us new networks at the leaves, networks that maximize the likelihood of the parents.

Fix a site, i.e., a character position in the sequence. Let l_i denote the length of the edge between leaf i and its parent, and b, c the value of a character at a leaf, chosen from set $S = \{0, 1\}$ in our case. Finally, let $p_{ab}(l)$ denote the substitution probability from character a to b along an edge of length l. For simplicity, assume that the given tree is binary (fully resolved); then *RefineML* proceeds as follows:

1. Learn the CPT parameters for the leaf networks reconstructed by the *DBN* method 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 current character b, set
   - $L_i(a) = \max_{c \in S} p_{ac}(l_i) \times Q_i(c)$
   - $C_i(a) = \arg\max_{c \in S} p_{ac}(l_i) \times Q_i(c)$

   where we have
   $$Q_i(c) = \begin{cases} k_b & \text{if } b = c \\ 1 - k_b & \text{otherwise} \end{cases} .$$
4. For each leaf i, assign its most likely character from the variable $C_i(a)$.

4. Experimental Design

The purpose of our experiments is to provide evidence for our hypothesis through a detailed examination of the sensitivity and specificity characteristics of our algorithm compared to the base inference algorithm. For such a purpose we need simulated data, as they allow us to control the parameters and, more importantly, to get an absolute assessment of accuracy. Other than possible issues about the biological verisimilitude of the simulated data, such simulations create the risk of introducing a systematic bias in the results. We take specific precautions against such bias, both in the design of the simulations and in the analysis.

4.1. Data

We generate test data from the phylogenetic tree, the network at the root, and the substitution matrix. Because reconstruction starts from gene-expression data, we not only create a collection of networks through simulated evolution, but we also compute CPTs for these networks and then use them to generate simulated gene-expression data. Fig. 1 illustrates the process; in the figure, the known conditions are shown with bold lines, characters in bold boxes, and the steps are labelled with italic characters. We begin by "evolving" networks down the tree, from the root network down, using the given substitution matrix and edge lengths—such simulated evolution is standard practice in the study of phylogenetic reconstruction [12, 16]. We use a wide variety of phylogenetic trees of modest sizes (from 20 to 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]. Our networks are of modest size, with 16 genes each; such networks make the gene-expression tables less "tall" and thus, in principle, less prone to generate errors in reconstruction, so that they present a more challenging case for our algorithm.

In order to get consistent CPTs, we assign weights to the edges of the networks, indicating how strong the relationships are. Therefore, in the data generation process, instead of using binary adjacency matrices, we use adjacency matrices with weights, chosen from the set $C = \{-3, -2, -1, 0, 1, 2, 3\}$. Absolute values show the strength of the edges, positive values denote reinforcement, negative values denote inhibition, and a 0 indicates the absence of an edge. The substitution matrix is now a 7×7 matrix. Weight values are assigned to the root network, yielding a weighted

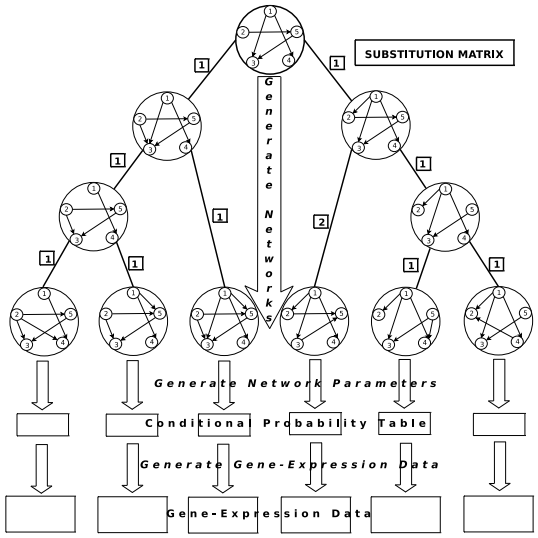


Figure 1. The data generation process

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 repeat as we traverse down the tree to obtain weighted adjacency matrices at the leaves.

We use binary gene expression levels, where 1 or 0 indicate that the gene is, respectively, *on* or *off*, as used in [15]. 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, let the expression levels of these nodes be denoted by the vector $Y_i = y_1 y_2 \cdots y_{m_i}$ and their confidence levels by the vector $W_i = w_1 w_2 \cdots w_{m_i}$. From Y_i and W_i, we can get the conditional probability $Pr(x_i|Y_i)$. Once we have the full parameters of the leaf networks, we use them to 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_i at time $t-1$ and the conditional probability $Pr(x_i|Y_i)$. We use 400 time points that is, the expression level matrix for each network has size 16×400.

Since the generation process is random according to the substitution probabilities and CPTs, we run it 15 times for each choice of tree structure and parameters. Of the 15 resulting datasets, 10 are left noiseless, while the other 5 have 10% noise added. Noiseless data should work better with the *DBN* model, which also makes it more difficult to improve on its results. For each fixed tree and root network, the mean and standard deviation of the results for both sets of datasets with and without noise are discussed.

4.2. 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. In order to verify this claim, we build a variation of *RefineML*, 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.

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

4.3. Measurements

On each dataset, we use the *DBN* model to predict regulatory networks with different penalty coefficients, from 0 to 0.5, with an interval of 0.05. For each penalty coefficient, we apply *RefineML*, *RefineLocal*, and *RefineRandomTree* on the predicted networks. We measure specificity and sensitivity to evaluate the performance of the algorithms.

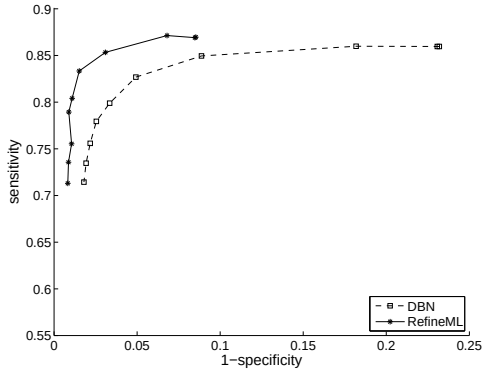
5. Results and Analysis

We present ROC curves for each algorithm, plotting *(1 − specificity)* vs. *sensitivity*, as measured for various penalty coefficients. In such plots, the closer a curve is to the top left-hand corner of the coordinate space, the better the results are. Space constraints prevent us from showing more than a sample of our results, so we show results on two representative trees: tree *T1* has 51 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 1.72 events (gain or loss of a regulatory arc in the network) per edge and resulting networks have from 23 to 36 edges. In the last figure, we present results of *DBN* and *RefineML* under different expected evolutionary rates for a third tree, *T3*, with 41 nodes on 6 levels.

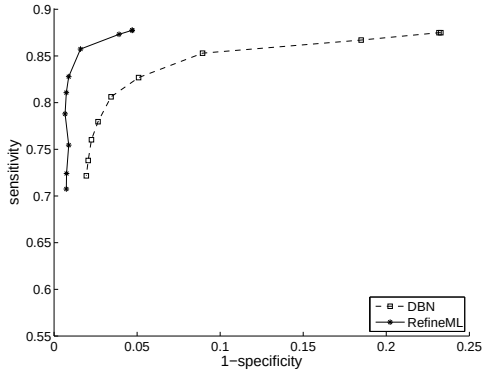
5.1. On boosting DBN

Fig. 2 shows the average performance of *RefineML* and *DBN* on 10 noiseless datasets. Throughout the range of parameters, our algorithm clearly dominates *DBN*. Not only does *RefineML* clearly improve the base algorithm, but it also shows very little loss of specificity (going from over 99% down to 91% on *T1* and 95% on *T2*) as sensitivity increases, in sharp contrast to *DBN*, where the curve flattens quickly for a decrease in specificity from a high of 98% down to 75%. Points at the bottom left-hand corner of the figures correspond to high penalty coefficients, where *DBN* has high specificity, but poor sensitivity; the corresponding values for our algorithms are farther left, indicating that specificity is even better, occasionally at a little expense in sensitivity (Fig. 2(b)). We also compared the average performance of *RefineML* and *DBN* on noisy datasets, where we found much the same picture. Actually our refinement algorithm yields more improvement on the noisy datasets, which are closer to the real data and thus cause more difficulties for *DBN* methods, yielding a larger margin for improvement. We choose to show results for noiseless datasets only, as the improvement caused by our algorithms can only increase as the noise level in the data increases.

The sample standard deviations (SD) of sensitivity and specificity for the 2 methods on the noiseless datasets are shown in Fig. 3 for both trees, using ellipses to denote the locus of one standard deviation around each point. (Note the change of scale from the preceding two figures to focus

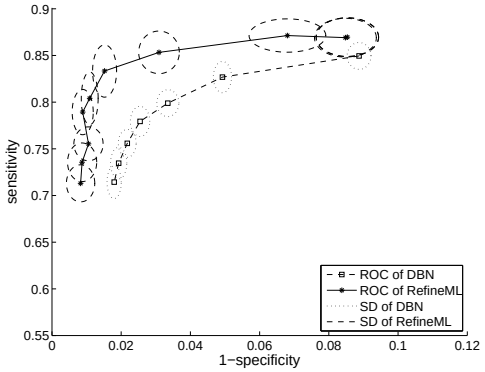


(a) ROC curves on tree *T1*

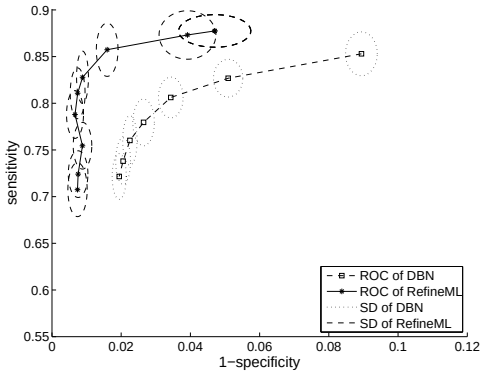


(b) ROC curves on tree *T2*

Figure 2. ROC curves for RefineML and DBN on noiseless datasets



(a) ROC curves and SD ellipses on tree *T1*



(b) ROC curves and SD ellipses on tree *T2*

Figure 3. ROC curves for RefineML and DBN with SDs on noiseless datasets

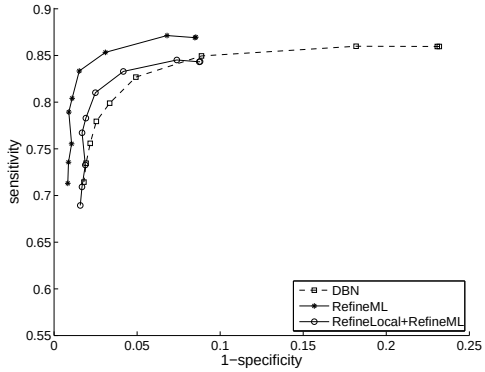
on higher specificity.) The curves are always at least 1 SD apart, so that our our method dominates the *DBN* method, not just on average, but also in the vast majority of cases.

5.2. On applying ML globally

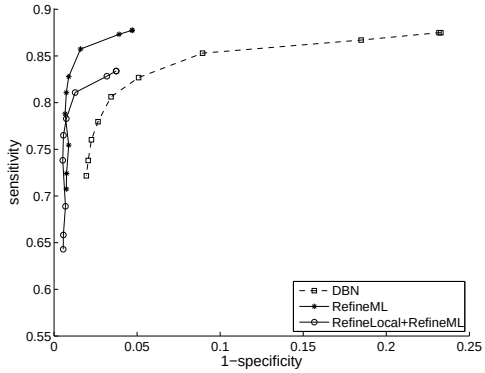
We now use *RefineLocal*, a variant of *RefineML* that infers ancestral networks only for the part of the tree that is used in the refinement phase, to show that the improvement wrought in the leaves by *RefineML* uses the phylogenetic information of the whole tree, not just the information present in the subforest induced by direct parents of leaves. Fig. 4 compares the performance of *RefineML* with that of its localized version on noiseless datasets. The plots are very similar: *RefineLocal* is clearly worse than the original algorithms, especially in terms of sensitivity. However, *RefineLocal* based on *RefineML* still outperforms *DBN*, due to the fact that *RefineML* maintains useful information from the leaves, though the ancestral inference procedure introduces significant additional errors when limited to small subtrees.

5.3. On phylogenetic information

We now use *RefineRandomTree*, a variation that carries out our full algorithm, but on a tree where the leaves have been reshuffled randomly to demonstrate the importance of correct phylogenetic information in the refinement procedure. Fig. 5 compares the performance of *RefineML* run on the correct phylogenetic tree with the average performance (over 100 runs) of *RefineRandomTree*. Again the contribution of the original networks at the leaves, as a function of the confidence values on the edges, compensates in part for the poor ancestral inference, so that the average performance of *RefineRandomTree* is typically better than that of *DBN* at higher sensitivity, although worse at higher specificity. (If we omit confidence values from *RefineRandomTree*, its performance becomes worse than that of *DBN*.) Overall, the results demonstrate both the value of

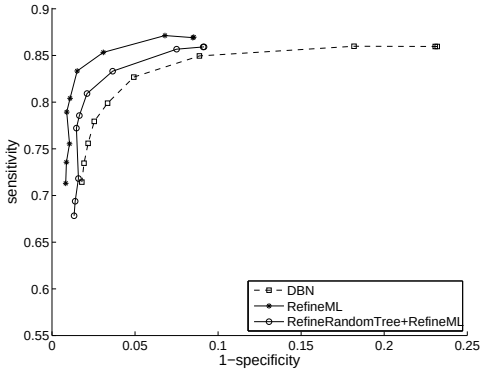


(a) ROC curves on tree $T1$

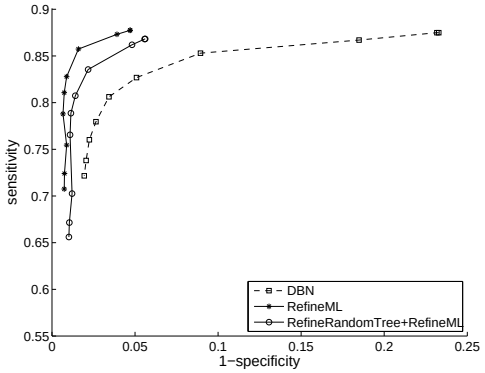


(b) ROC curves on tree $T2$

Figure 4. ROC curves for RefineML and RefineLocal



(a) ROC curves on tree $T1$



(b) ROC curves on tree $T2$

Figure 5. ROC curves for RefineML run on the correct tree and on random trees

correct phylogenetic data and the value of the information present in the original leaf networks.

5.4. On different evolutionary rates

The expected evolutionary rate (average edge length) was fixed in all experiments presented above. High rates of evolution cause *saturation*, where the apparent number of evolutionary changes needed to explain the observed differences underestimates the actual number of changes that occurred over time. This problem is aggravated when each character has only two states, as two changes to the same character cancel each other. We thus expect our method to become less effective as evolutionary rates increase.

In order to study this problem, we conducted experiments on tree $T3$ of 21 leaves with a fixed root network of 16 nodes and 24 edges, using different evolutionary rates to generate the leaf networks. Fig. 6 shows ROC curves for *RefineML* and *DBN* with evolutionary rates of 2.6, 4.4 and 6.6 on noiseless datasets. The loss in performance as the rate of evolution increases is clear for both methods; since *DBN* itself suffers (perhaps because some networks produced in the simulation violate implicit assumptions used in *DBN*), the loss in performance of *RefineML* is a combination of worsened leaf networks returned by *DBN* and worsened ancestral reconstruction by FastML. However, performance is good at the already high evolutionary rate of 2.6: in $T3$, most paths from the root to a leaf have 5 edges and thus have, with a rate of 2.6, an expected length of 13, meaning that leaf networks generated during the simulation will differ from the root network by about 13 changes, out of a total of 24 edges. At the highest rate of evolution shown, the number of changes between two leaf networks is comparable to the numbers of their edges. These are huge numbers for phylogenetic reconstruction: it is thus gratifying that our method remains able to boost the base method.

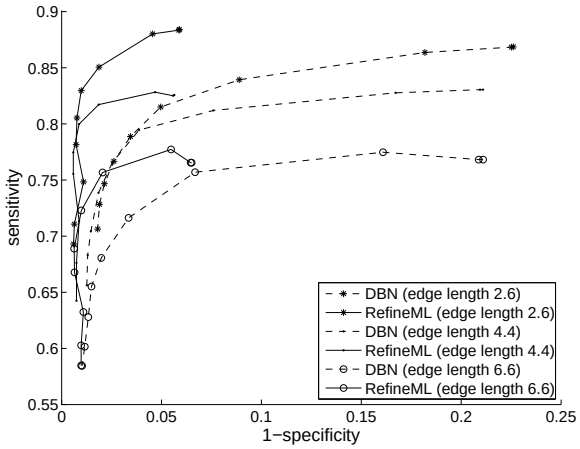


Figure 6. ROC curves for DBN and RefineML under different expected evolutionary rates

6. Conclusions and Future Work

We gave experimental evidence to support our claim that phylogenetic information can be used to improve the inference of regulatory networks for a family of related organisms and described algorithms to exploit this information. Our simulation studies demonstrate clear improvements in specificity and sensitivity. Our approach is best viewed as a booster for existing inference algorithms and can, in principle, be used with any favorite network inference tool and any favorite phylogenetic reconstruction algorithm.

While our work offers a novel way to infer regulatory networks with high accuracy, much remains to be done. Our data generation, while corrupted by a fair amount of noise, is well matched to the model conditions assumed by our algorithms; biological data may prove more difficult to handle. Our approach requires comparable gene-expression data for a number of closely related organisms; cost and technical feasibility are not an issue, but ensuring comparability remains challenging. Finally, our approach is experimental and needs to be placed on a solid mathematical foundation.

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 Biocomp. (PSB'99)*, 17–28. World Scientific, 1999.

[2] M. Babu et al., Structure and evolution of transcriptional regulatory networks. *Curr. Opinion in Struct. Bio.*, 14(3):283–291, 2004.

[3] M. Babu and S. Teichmann. Evolution of transcription factors and the gene regulatory network in Escherichia coli. *Nucleic Acids Res.*, 31(4):1234–1244, 2003.

[4] M. Babu, S. 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] T. Chen, H. He, and G. Church. Modeling gene expression with differential equations. In *Proc. 4th Pacific Symp. on Biocomp. (PSB'99)*, 29–40. World Scientific, 1999.

[7] R. Conant. Extended dependency analysis of large systems. *Int'l J. General Systems*, 14(2):97–141, 1988.

[8] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comp. Bio.*, 9:67–103, 2002.

[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. Comp. Bio.*, 7:601–620, 2000.

[11] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proc. 14th Conf. on Uncertainty in Art. Intell. UAI'98*, 139–147, 1998.

[12] D. Hillis. Approaches for assessing phylogenetic accuracy. *Syst. Bio.*, 44:3–16, 1995.

[13] M. Kanehisa et al., From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.*, 34:D354–D357, 2006.

[14] S. 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 Biocomp. (PSB'98)*, 18–29. World Scientific, 1998.

[16] B. Moret and T. Warnow. Reconstructing optimal phylogenetic trees: A challenge in experimental algorithmics. In R. Fleischer, B. Moret, and E. Schmidt, eds., *Experimental Algorithmics*, vol. 2547 of *Lecture Notes in Computer Science*, 163–180. Springer Verlag, 2002.

[17] K. Murphy. The Bayes net toolbox for MATLAB. *Comp. Sci. & Statistics*, 33:331–351, 2001.

[18] K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, U. California, Berkeley, 1999. `www.csail.mit.edu/~murphyk/Papers/ismb99.ps.gz`.

[19] 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.

[20] D. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature Genetics*, 32:502–508, 2002.

[21] S. Teichmann and M. Babu. Gene regulatory network growth by duplication. *Nature Genetics*, 36(5):492–496, 2004.

[22] R. Xu, X. Hu, and D. Wunsch. Inference of genetic regulatory networks from time series gene expression data. In *Proc. IEEE Int'l Joint Conf. on Neural Networks*, vol. 2, 1215–1220. IEEE Press, Piscataway, NJ, 2004.

[23] W. Zhao, E. Serpedin, and E. Dougherty. Inferring gene regulatory networks from time series data using the minimum length description principle. *Bioinformatics*, 22(17):2129–2135, 2006.