Improving Inference of Transcriptional Regulatory Networks Based on Network Evolutionary Models

Xiuwei Zhang and Bernard M.E. Moret

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

Abstract. Computational inference of transcriptional regulatory networks remains a challenging problem, in part due to the lack of strong network models. In this paper we present evolutionary approaches to improve the inference of regulatory networks for a family of organisms by developing an evolutionary model for these networks and taking advantage of established phylogenetic relationships among these organisms. In previous work, we used a simple evolutionary model for regulatory networks and provided extensive simulation results showing that phylogenetic information, combined with such a model, could be used to gain significant improvements on the performance of current inference algorithms.
In this paper, we extend the evolutionary model so as to take into account gene duplications and losses, which are viewed as major drivers in the evolution of regulatory networks. We show how to adapt our evolutionary approach to this new model and provide detailed simulation results, which show improvement both on the reference network inference algorithms and on the results obtained under the simpler model. We also provide results on biological data (cis-regulatory modules for 12 species of *Drosophila*), confirming our simulation results.

1 INTRODUCTION

Transcriptional regulatory networks are models of the cellular regulatory system that governs transcription. Because establishing the topology of the network from bench experiments is very difficult and time-consuming, regulatory networks are commonly inferred from gene-expression data. Various computational models, such as Boolean networks [1], Bayesian networks [11], dynamic Bayesian networks (DBNs) [17], and differential equations [8, 27], have been proposed for this purpose, along with associated inference algorithms. Results, however, have proved mixed: the high noise level in the data, the paucity of well studied networks, and the many simplifications made in the models all combine to make inference difficult.

Bioinformatics has long used comparative and, more generally, evolutionary approaches to improve the accuracy of computational analyses. Work by Babu's group [3, 4, 26] on the evolution of regulatory networks in *E. coli* and *S. cerevisiae* has demonstrated the applicability of such approaches to regulatory networks. They posit a simple evolutionary model for regulatory networks, under which network edges are simply added or removed; they study how well such a model accounts for the dynamic evolution of the two most studied regulatory networks; they then investigate the evolution

of regulatory networks with gene duplications [26], concluding that gene duplication plays a major role, in agreement with other work [23].

Phylogenetic relationships are well established for many groups of organisms; as the regulatory networks evolved along the same lineages, the phylogenetic relationships informed this evolution and so can be used to improve the inference of regulatory networks. Indeed, Bourque and Sankoff [7] developed an integrated algorithm to infer regulatory networks across a group of species whose phylogenetic relationships are known; they used the phylogeny to reconstruct networks under a simple parsimony criterion. In previous work [29], we presented two refinement algorithms, both based on phylogenetic information and using a likelihood framework, that boost the performance of any chosen network inference method. On simulated data, the *receiver-operator characteristic (ROC)* curves for our algorithms consistently dominated those of the standard approaches used alone; under comparable conditions, they also dominated the results from Bourque and Sankoff. Both our previous approach and that of Bourque and Sankoff are based on an evolutionary model that considers only edge gains and losses, so that the networks must all have the same number of genes (orthologous across all species). Moreover, the gain or loss of an edge in that model is independent of any other events. However, this process accounts for only a small part of regulatory network evolution; in particular, gene duplication is known to be a crucial source of new genetic function and a mechanism of evolutionary novelty [23, 26].

In this paper we present a model of network evolution that takes into account gene duplications and losses and their effect on regulatory network structures. Such a model provides a direct evolutionary mechanism for edge gains and losses, while also enabling broader application and more flexible parameterization. To refine networks using phylogenetic information within this broader framework, we use the reconciliation of gene trees with the species tree [2, 10, 21] to process gene duplication and loss events. We then extend our refinement algorithms [29]; thanks to the more complex model, these refinement algorithms are more broadly applicable, while also returning better solutions. We provide experimental results confirming that our new algorithms provide significant improvements over the base inference algorithms, while also improving over our refinement algorithms using the simpler model of network evolution.

2 BACKGROUND

Our refinement algorithms [29] work iteratively in two phases after an initialization step. First, we obtain the regulatory networks for the family of organisms; typically, these networks are inferred from gene-expression data for these organisms, using standard inference methods. We place these networks at the corresponding leaves of the phylogeny of the family of organisms and encode them into binary strings by simply concatenating the rows of their adjacency matrix. We then enter the iterative refinement cycle. In the first phase, we infer ancestral networks for the phylogeny (strings labelling internal nodes), using our own adaptation of the FastML [22] algorithm; in the second phase, these ancestral networks are used to refine the leaf networks. These two phases are then repeated as needed. Our refinement algorithms are formulated within a maximum likelihood (ML) framework and focused solely on refinement—they are algorithmic boosters for one's preferred network inference method. Our new algorithms

retain the same general approach, but include many changes to use the duplication/loss data and handle the new model.

2.1 Base network inference methods

We use both DBN and differential equation models as base inference methods in our experiments. When DBNs are used to model regulatory networks, an associated structure learning algorithm is used to infer the networks from gene-expression data [12, 17, 18]; so as to avoid overly complex networks, a penalty on graph structure complexity is usually added to the ML score, thereby reducing the number of false positive edges. In [29] we used a coefficient k_p to adjust the weight of this penalty and study different tradeoffs between sensitivity and specificity, yielding the optimization criterion $\log Pr(D|G,\hat{\Theta}_G) - k_p \#G \log N$, where D denotes the dataset used in learning, G is the (structure of the) network, $\hat{\Theta}_G$ is the ML estimate of parameters for G, $\#G$ is the number of free parameters of G, and N is the number of samples in D.

In models based on differential equations [8, 27], a regulatory network is represented by the equation system $dx/dt = f(x(t)) - Kx(t)$, 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 $f(\cdot)$. To get networks with different levels of sparseness, we applied different thresholds to the connection matrix to get final edges.

In our experiments we use Murphy's Bayesian Network Toolbox [20] for the DBN approach and TRNinfer [27] for the differential equation approach; we refer to them as *DBI* and *DEI*, respectively.

2.2 Refinement algorithms in our previous work

The principle of our phylogenetic approach is that phylogenetically close organisms are likely to have similar close regulatory networks; thus independent network inference errors at the leaves get corrected in the ancestral reconstruction process along the phylogeny. We gave two refinement algorithms, *RefineFast* and *RefineML*. Each uses the globally optimized parents of the leaves to refine the leaves, but the first simply picks a new leaf network from the inferred distribution (given the parent network, the evolutionary model parameters, and the phylogeny), while the second combines the inferred distribution with a prior, the existing leaf network, using a precomputed *belief coefficient* that indicates our confidence in the current leaf network, and returns the most likely network under these parameters.

2.3 Reconciliation of species tree and gene trees

To infer ancestral networks with the extended network evolution model, we need a full history of gene duplications and losses. We reconstruct this history by reconciling the gene trees and the species tree. The species tree is the phylogenetic tree whose leaves correspond to the modern organisms; gene duplications and losses occur along the branches of this tree. A gene tree is a phylogenetic tree whose leaves correspond to genes in orthologous gene families across the organisms of interest; in such a tree, gene duplications and speciations are associated with internal nodes.

When gene duplications and losses occur, the species trees and the gene trees may legitimately differ in topology. Reconciling these superficially conflicting topologies—

that is, explaining the differences through a history of gene duplications and losses—is known as *lineage sorting* or *reconciliation* and produces a list of gene duplications and losses along each edge in the species tree. While reconciliation is a hard computational problem, algorithms have been devised for it in a Bayesian framework [2] or using a simple parsimony criterion [10].

3 THE NEW NETWORK EVOLUTION MODEL

Although transcriptional regulatory networks produced from bench experiments are available for only a few model organisms, other types of data have been used to assist in the comparative study of regulatory mechanisms across organisms [9, 24, 25]. For example, gene-expression data [25], sequence data like transcriptional factor binding site (TFBS) data [9, 24], and *cis*-regulatory elements [25] have all been used in this context. Moreover, a broad range of model organisms have been studied, including bacteria [4], yeast [9, 25], and fruit fly [24], thus spanning a broad evolutionary range. However, while these studies are promising, they have not to date sufficed to establish a clear model for regulatory networks or their evolution.

Our new model remains simple, but can easily be generalized or associated with other models, such as the evolutionary model of TFBSs [16]. In this new model, the networks are represented by binary adjacency matrices. The evolutionary operations are:

- *Gene duplication*: a gene is duplicated with probability p_d. After a duplication, edges for the newly generated copy can be assigned as follows:
  - *Neutral initialization:* Create connections between the new copy and other genes randomly according to the proportion π_1 of edges in the background network independently of the original copy.
  - *Inheritance initialization:* Connections of the duplicated copy are reported to correlate with those of the original copy [4, 23, 26]. This observation suggests letting the new copy inherit the connections of the original, then lose some of them or gain new ones at some fixed rate [6].
  - *Preferential attachment:* The new copy gets preferentially connected to genes with high connectivity [5, 6].
- *Gene loss*: a gene is deleted along with all its connections with probability p_l.
- *Edge gain*: an edge between two genes is generated with probability p_{01}.
- *Edge loss*: an existing edge is deleted with probability p_{10}.

The model parameters are thus p_d, p_l, the proportions of 0s and 1s in the networks $\Pi = (\pi_0\ \pi_1)$, the substitution matrix of 0s and 1s, $P = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix}$, plus parameters suitable to the initialization model.

4 THE NEW REFINEMENT METHODS

We begin by collecting the regulatory networks to be refined. These networks may have already been inferred or they can be inferred from gene-expression data at this point using any of the standard network inference methods introduced in Sec. 2.1. The genes in these networks are not required to be orthologous across all species, as the duplication/loss model allows for gene families of various sizes. Refinement proceeds in the

two-phase iterative manner already described, but adding a step for lineage reconciliation and suitably modified algorithms for ancestral reconstruction and leaf refinement:

1. Reconstruct the gene trees, one for each gene family.
2. Reconstruct the history of gene duplications and losses by reconciling these gene trees and the given species tree. From this history, determine gene contents for the ancestral regulatory networks (at each internal node of the species tree).
3. Infer the edges in the ancestral networks. We do this using a revised version of FastML, described in Sec. 4.1.
4. Refine the leaf networks with new versions of *RefineFast* and *RefineML*, described in Secs. 4.2 and 4.3.
5. Repeat steps 3 and 4 as needed.

4.1 Inferring ancestral networks

FastML assumes independence among the entries of the adjacency matrices and reconstructs ancestral characters one site at a time. When the gene content is the same in all networks, FastML can be used nearly as such. In our new model, however, the gene content varies across networks. We solve this problem by embedding all networks into one that includes every gene that appears in any network, taking the union of all gene sets. We then represent a network with a ternary matrix, with the special character x used to denote a missing gene. All networks are thus represented with adjacency matrices of the same size. Since the gene contents of ancestral networks are known thanks to reconciliation, the entries with x are already identified in their matrices; other entries are reconstructed by our revised version of FastML, with a new character set $S' = \{0,1,x\}$. We modify the substitution matrix and take special measures for x during calculation. The substitution matrix P' for S' can be derived from the model parameters in Sec. 3, without introducing new parameters.

$$P' = \begin{pmatrix} p'_{00} & p'_{01} & p'_{0x} \\ p'_{10} & p'_{11} & p'_{1x} \\ p'_{x0} & p'_{x1} & p'_{xx} \end{pmatrix} = \begin{pmatrix} (1-p_l)\cdot p_{00} & (1-p_l)\cdot p_{01} & p_l \\ (1-p_l)\cdot p_{10} & (1-p_l)\cdot p_{11} & p_l \\ p_d \cdot \pi_0 & p_d \cdot \pi_1 & 1-p_d \end{pmatrix}$$

Given P', let i, j, k denote a tree node, and $a,b,c \in S'$ possible values of a character at some node. 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 for i, given that its parent is assigned character a.

On a binary phylogenetic tree, for each site, the revised FastML then works as follows:

1. If leaf i has character b, then, for each $a \in S'$, set $C_i(a) = b$ and $L_i(a) = p'_{ab}$.
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
   - if i has character x, for each $a \in S'$, set $L_i(a) = p'_{ax} \cdot L_j(x) \cdot L_k(x)$ and $C_i(a) = x$;
   - otherwise, for each $a \in S'$, set $L_i(a) = \max_{c\in\{0,1\}} p'_{ac} \cdot L_j(c) \cdot L_k(c)$ and $C_i(a) = \arg\max_{c\in\{0,1\}} p'_{ac} \cdot L_j(c) \cdot L_k(c)$.
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 $a \in \{0,1\}$ that maximizes $\pi_a \cdot L_j(a) \cdot L_k(a)$, if the character of i is not already identified as x.
5. Traverse the tree from the root, assigning to each node its character by $C_i(a)$.

4.2 Refining leaf networks: RefineFast

RefineFast uses the parent networks inferred by FastML to evolve new sample leaf networks. Because the strategy is just one of sampling, we do not alter the gene contents of the original leaves—duplication and loss are not taken into account in this refinement step. Let A_l and A_p be the adjacency matrices of a leaf network and its parent network, respectively, and let A'_l stand for the refined network for A_l; then the revised *RefineFast* algorithm carries out the following steps:

1. For each entry (i, j) of each leaf network A_l,
   - if $A_l(i, j) \neq x$ and $A_p(i, j) \neq x$, evolve $A_p(i, j)$ by P to get $A'_l(i, j)$;
   - otherwise, assign $A'_l(i, j) = A_l(i, j)$.
2. Use the $A'_l(i, j)$ to replace $A_l(i, j)$.

In this algorithm, the original leaf networks are used only in the first round of ancestral reconstruction, after which they are replaced with the sample networks drawn from the distribution of possible children of the parents.

4.3 Refining leaf networks: RefineML

To make use of the prior information (the original networks), *RefineML* uses a *belief coefficient* k_b for each edge of these networks. In the DBN framework, these coefficients can be calculated from the *conditional probability table* (CPT) parameters of the predicted networks.

As for *RefineFast*, the refinement procedure does not alter the gene contents of the leaves. Using the same notations as in Sec. 4.1 and 4.2, *RefineML* aims to find the A'_l which maximizes the likelihood of the subtree between A_p and A'_l. The revised *RefineML* algorithm thus works 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. For each entry (i, j) of each leaf network A_l, do:
   - If $A_l(i, j) \neq x$ and $A_p(i, j) \neq x$, let $a = A_p(i, j)$, $b = A_l(i, j)$,
     (a) let $Q(c) = k_b$ if $b = c$, $1 - k_b$ otherwise;
     (b) calculate the likelihood $L(a) = \max_{c \in \{0,1\}} p_{ac} \cdot Q(c)$;
     (c) assign $A'_l(i, j) = \arg\max_{c \in \{0,1\}} p_{ac} \cdot Q(c)$.
   - Otherwise, assign $A'_l(i, j) = A_l(i, j)$.
3. Use $A'_l(i, j)$ to replace $A'_l(i, j)$.

5 EXPERIMENTAL DESIGN

To test the performance of our approach, we need regulatory networks as the input to our refinement algorithms. In our simulation experiments, we evolve networks along a given tree from a chosen ancestral network to obtain the "true" leaf networks. Then, in order to reduce the correlation between generation and reconstruction of networks, we use the leaf networks to create simulated expression data and use our preferred network inference method to reconstruct networks from the expression data. These inferred networks are the true starting point of our refinement procedure—we use the simulated gene expression data only to achieve better separation between the generation of networks and their refinement, and also to provide a glimpse of a full analysis pipeline

for biological data. We then compare against the "true" networks (generated in the first step) the inferred networks after and before refinement.

Despite of the advantages of such simulation experiments (which allow an exact assessment of the performance of the inference and refinement algorithms), results on biological data are highly desirable, as such data may prove quite different from what was generated in our simulations. TFBS data is used to study regulatory networks, assuming that the regulatory interactions determined by transcription factor (TF) binding share many properties with the real interactions [9, 13, 24]. Given this close relationship between regulatory networks and TFBSs and given the large amount of available data on TFBSs, we chose to use TFBS data to derive regulatory networks for the organisms as their "true" networks—rather than generate these networks through simulation. In this fashion, we produce datasets for the *cis*-regulatory modules (CRMs) for 12 species of *Drosophila*

With the extended evolutionary model, conducting experiments with real data involves several extra steps besides the refinement step, each of which is a source of potential errors. For example, assuming we have identified gene families of interest, we need to build gene trees for these genes so as to be able to reconstruct a history of duplications and losses. Any error in gene tree reconstruction leads to magnified errors in the history of duplications and losses. Assessing the results of data analysis under such circumstances (no knowledge of the true networks and many complex sources of error) is not possible, so we turned to simulation for this part of the testing. This decision does not prejudice our ability to apply our approach to real data and to infer high-quality networks: it only reflects our inability to compute precise accuracy scores on biological data.

5.1 Experiments on biological data with the basic evolutionary model

We use regulatory networks derived from TFBS data as the "true" networks for the organisms rather than generating these networks through simulations. Such data is available for the *Drosophila* family (whose phylogeny is well studied) with 12 organisms: *D. simulans*, *D. sechellia*, *D. melanogaster*, *D. yakuba*, *D. erecta*, *D. ananassae*, *D. pseudoobscura*, *D. persimilis*, *D. willistoni*, *D. mojavensis*, *D. virilis* and *D. grimshawi*. The TFBS data is drawn from the work of Kim *et al.* [16], where the TFBSs are annotated for all the 12 organisms on 51 CRMs.

We conduct separate experiments on different CRMs. For each CRM, we choose orthologous TFBS sequences of 6 transcription factors (TFs): *Dstat*, *Bicoid*, *Caudal*, *Hunchback*, *Kruppel* and *Tailless*, across the 12 organisms. Then for each organism, we can get a network with these 6 TFs and the target genes indicated by the TFBS sequences, where the arcs are determined by the annotation of TFBSs, and the weights of arcs are calculated from the binding scores provided in [16]. (In this paper we do not distinguish TFs and target genes and call them all "genes.") These networks are regarded as the "true" regulatory networks for the organisms.

Gene-expression data is then generated from these "true" regulatory networks; data is generated independently for each organism, using procedure *DBNSim*, based on the DBN model [29]. Following [18], *DBNSim* uses 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 $Pr(x_i|y)$.

We generate 100 time points of gene-expression data for each network in this manner. With this data we can apply the approach in Sec. 4. *DBI* is applied to infer regulatory networks from the gene-expression data. The inferred networks are then refined by *RefineFast* and *RefineML*. The whole procedure is run 10 times to provide smoothing and we report average performance over these runs.

5.2 Experiments on simulated data with the extended model

In these experiments, the "true" networks for the organisms and their gene-expression data are both generated, starting from three pieces of input information: the phylogenetic tree, the network at the root, and the evolutionary model. While simulated data allows us to get absolute evaluation of our refinement algorithms, specific precautions need to be taken against systematic bias during data simulation and result analysis. 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 [15], as also used by Kim *et al.* [17]; we also explore a wide range of evolutionary rates, especially different rates of gene duplication and loss. The root network is of modest size, between 14 and 17 genes, a relatively easy case for inference algorithms and thus also a more challenging case for a boosting algorithm.

We first generate the leaf networks that are used as the "true" regulatory networks for the chosen organisms. Since we need quantitative relationships in the networks in order to generate gene-expression data from each network, 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_p. To get the adjacency matrix for its child A_c, according to the extended network evolution model, we follow two steps: evolve the gene contents and evolve the regulatory connections. First, genes are duplicated or lost by p_d and p_l. If a duplication happens, a row and column for this new copy will be added to A_p, the values initialized either according to the *neutral initialization* model or the *inheritance initialization* model. (We conducted experiments under both models.) We denote the current adjacency matrix as A'_c. Secondly, edges in A'_c are mutated according to P_{01} and p_{10} to get A_c. We repeat this process as we traverse down the tree to obtain weighted adjacency matrices at the leaves, which is standard practice in the study of phylogenetic reconstruction [14, 19].

Simulation experiments allow us to record the real gene duplication and loss history during data generation, so that we can test the pure accuracy of the refinement algorithms, without mixing their performance with that of gene tree reconstruction or reconciliation.

To test our refinement algorithms on different kinds of data, besides *DBNSim*, we also use Yu's GeneSim [28] to generate gene-expression data from the weighted networks. GeneSim [28] can produce simulated continuous gene-expression values for a

given weighted network as well as generate arbitrary network structures. 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 $x(t+1) = x(t) + k_e(x(t) - z)C + \varepsilon$.

With two data generation methods, *DBNSim* and GeneSim, and two base inference 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 *DBNSim* to generate data for *DBI*. $13 \times n$ time points are generated for a network with n genes, since larger networks generally need more samples to gain comparable inference accuracy to smaller ones. Second, we apply *DEI* to datasets generated by GeneSim to infer the networks. Since *DEI* does not accept large datasets (with many time points), here we use smaller datasets than the previous group of experiments with at most 75 time points. For each setup, experiments with different gene duplication and loss rates are conducted, and each experiment is run 10 times to obtain average performance.

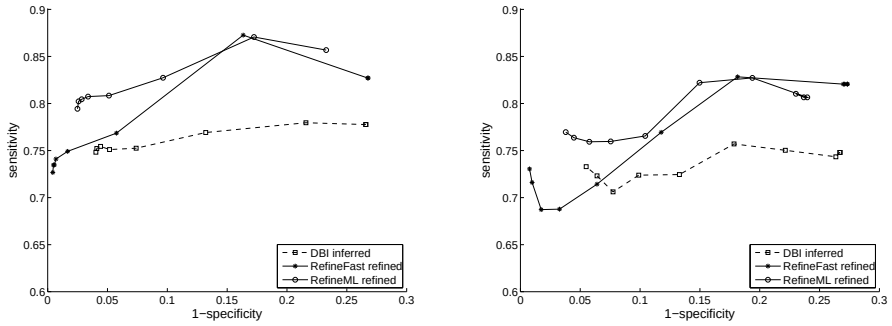
5.3 Measurements

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* and *RefineML* 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* 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

6.1 Results on biological data with basic evolutionary model

Experiments were conducted on different CRMs of the 12 *Drosophila* species; here we show results on two of them. In both experiments, regulatory networks have 6 TFs and 12 target genes, forming networks with 18 nodes. Average performance for the base inference algorithm (*DBI*) and for the two refinement algorithms over 8 runs for these two experiments is shown in Fig. 1 using ROC curves. In the two plots, the points on each curve are obtained with different structure complexity penalty coefficients. From Fig. 1 we can see the improvement of our refinement algorithms over the base algorithm is significant: *RefineML* improves significantly both sensitivity and specificity,



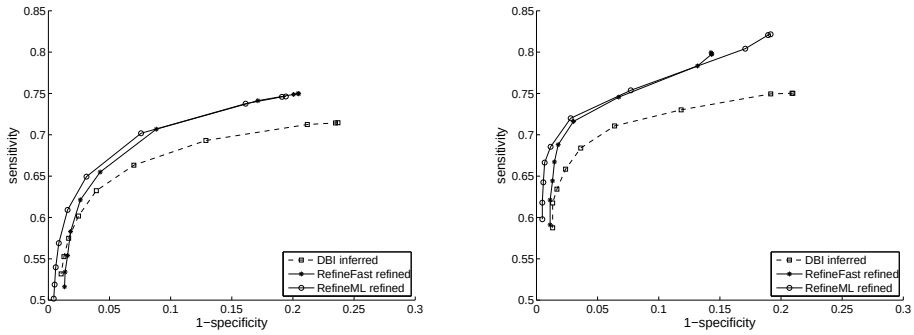
(a) *Results on CRM abd-A_iab-2_1.7_* (b) *Results on CRM Abd-B_IAB5*

Fig. 1. Performance of refinement algorithms on *Drosophila* data

while *RefineFast* loses a little sensitivity while gaining more specificity for sparse networks. The dominance of *RefineML* over *RefineFast* shows the advantage of reusing the leaf networks inferred by base algorithms, especially when the error rate in these leaf networks is low. Results on other CRMs also show similar improvement of our refinement algorithms.

Besides the obvious improvement, we can also observe the fluctuation of the curves: theoretically sensitivity can be traded for specificity and vice versa, so that the ROC curves should be in the shapes similar to those in Fig. 2, but the curves in the figure are not as well shaped. Various factors can account for this: the shortage of gene-expression data to infer the network, uncertainty and noise of biological data, the special structure of the networks to be inferred, or the relatively small amount of data involved, leading to higher variability. (We have excluded the first possibility by generating larger gene-expression datasets for inference algorithms, where similar fluctuations still occur.)

We analyze the difference level between the "true" networks of the 12 organisms, to obtain a view of the evolutionary rate of regulatory networks in our datasets. For



(a) *Results with higher gene duplication and loss rates* (b) *Results with lower gene duplication and loss rates*

Fig. 2. Performance with extended evolution model and *DBI* inference method

each CRM, we take the union of the edges in all 12 networks, classify these edges by the number of networks in which they are present, and calculate the proportion of each category. The overall proportions on all CRMs shown in Table 1, where we see there is a big fraction of edges which are shared by less than half organisms, meaning that the networks are quite diverse. This observation shows the improvement brought by the refinement algorithms is due to proper use of phylogenetic information rather than the averaging effect of trees.

Table 1. The proportion of edges shared by different number of species

1	2	3	4	5	6	7	8	9	10	11	12
0.19	0.18	0.03	0.07	0.03	0.09	0.03	0.09	0.02	0.07	0.07	0.13

6.2 Results with extended evolutionary model

Simulation experiments on various combinations of gene-expression data generation methods and network inference methods were conducted. All results we show below are on one representative phylogenetic tree with 37 nodes on 7 levels. Since the results of using *neutral initialization* model and *inheritance initialization* model in data generation are very similar, we only show results with the *neutral initialization* model.

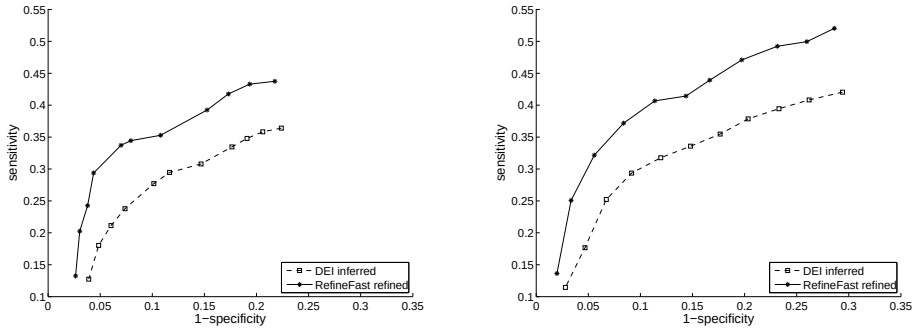
With *DBI* as the base inference algorithm

In this series of experiments, *DBNSim* is used to generate gene-expression data, and *DBI* as base inference algorithm. The root network has 16 genes. Fig. 2 shows the average performance of these algorithms over 10 runs. In [29] we tested different evolutionary rates, which were mainly edge gain or loss rates; here we focus on testing different gene duplication and loss rates. Fig. 2(a) shows results under a relatively high rate of gene duplication and loss (resulting in 32 gene duplication and 25 loss events along the tree), while Fig 2(b) has a lower rate (with 15 gene duplication events and 7 gene loss events), again averaged over 10 runs. Given the size of the tree, these are high rates of duplication and loss, yet, as we can see from Fig. 2, the improvement gained by our refinement algorithms remains clear, with Fig. 2(b) showing slightly more improvement than Fig. 2(a), especially for sensitivity.

Comparing Fig. 1 and 2, one can observe that in Fig. 1 *RefineML* does better than *RefineFast*, but in Fig. 2 they have comparable performance. The main reason is that, as described in Section 4, *RefineML* benefits from the useful information in the *DBI* inferred networks, and bad performance of *DBI* provides little advantage for *RefineML* over *RefineFast*.

With *DEI* as the base inference algorithm

Here GeneSim is used to generate continuous gene-expression data for network inference with *DEI*. The root network has 14 genes. We also show results for 2 experiments: Fig. 3(a) has higher gene duplication and loss rates, resulting in 15 gene duplications and 7 gene losses, while datasets in Fig. 3(b) have an average of 8 gene duplications and 3 losses. The *DEI* tool that we use, aims to infer networks with small



(a) *Results with higher gene duplication and loss rates* (b) *Results with lower gene duplication and loss rates*

Fig. 3. Performance with extended evolution model and *DEI* inference method

gene-expression datasets. Fig. 3 shows the average performance of *DEI* and *RefineFast* for both experiments over 10 runs. *RefineFast* improves much the performance of the base algorithm, especially the sensitivity, which is poorly done by *DEI* in both plots. Again more improvement is observed with the experiment which has less gene duplications and losses. This is because high duplication and loss rates give rise to a large overall gene population, yet many of them exist only in a few leaves, so that there is not much phylogenetic information to be used to correct the prediction of the connections for these genes.

7 CONCLUSIONS AND FUTURE WORK

We presented a model, associated algorithms, and experimental results to test the hypothesis that a more refined model of transcriptional regulatory network evolution would support additional refinements in accuracy. Specifically, we presented a new version of our evolutionary approach to refine the accuracy of transcriptional regulatory networks for phylogenetically related organisms, based on an extended network evolution model, which takes into account gene duplication and loss. Modelling these events is important because gene duplication is known to be a source of new genetic function and a mechanism of evolutionary novelty [23, 26]. Results of experiments under various settings show the effectiveness of our refinement algorithms with the new model throughout a broad range of gene duplications and losses.

We also collected regulatory networks from the TFBS data of 12 *Drosophila* species and applied our approach (using the basic model), with very encouraging results. These results confirm that phylogenetic relationships carry over to the regulatory networks of a family of organisms and can be used to improve the network inference and to help with further analysis of regulatory systems and their dynamics and evolution.

Our positive results with the extended network evolution model show that refined models can be used in inference to good effect. Our current model can itself be refined, by using the widely studied evolution of TFBS [16, 24].

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. L. Arvestad, A.-C. Berglund, J. Lagergren and B. Sennblad. Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In *Proc. 8th Conf. Research in Comput. Mol. Bio. RECOMB'04*, pp. 326–335, 2004.
3. 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.
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. A.-L. Barabási and Z.N. Oltvai. Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* 5:101–113, 2004.
6. A. Bhan, D.J. Galas, and T.G. Dewey. A duplication growth model of gene expression networks. *Bioinformatics* 18(11):1486–1493, 2002.
7. G. Bourque and D. Sankoff. Improving gene network inference by comparing expression time-series across species, developmental stages or tissues. *J. Bioinform. Comput. Bio.* 2(4):765–783, 2004.
8. 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.
9. A. Crombach and P. Hogeweg. Evolution of evolvability in gene regulatory networks. *PLoS Comput. Bio.* 4(7):e1000112, 2008.
10. D. Durand, B.V. Halldórsson, and B. Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J. Comput. Bio.* 13(2):320–335, 2006.
11. 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.
12. 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.
13. C.T. Harbison, D.B. Gordon, T.I. Lee, *et al.* Transcriptional regulatory code of a eukaryotic genome. *Nature* 431:99–104, 2004.
14. D.M. Hillis. Approaches for assessing phylogenetic accuracy. *Syst. Bio.* 44:3–16, 1995.
15. M. Kanehisa *et al.* From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.* 34:D354–D357, 2006.
16. J. Kim, X. He, and S. Sinha. Evolution of regulatory sequences in 12 *Drosophila* species. *PLoS Genet.* 5(1):e1000330, 2009.
17. S.Y. Kim, S. Imoto, and S. Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinf.* 4(3):228–235, 2003.
18. 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.
19. 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, eds., *Experimental Algorithmics*, *LNCS* **2547** (2002), pp. 163–180.
20. K.P. Murphy. The Bayes net toolbox for MATLAB. *Comput. Sci. Stat.* 33:331–351, 2001.
21. R.D.M. Page and M.A. Charleston. From gene to organismal phylogeny: Reconciled trees and the gene tree/species tree problem. *Mol. Phyl. Evol.* 7(2):231–240, 1997.
22. 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.

23. C. Roth *et al.* Evolution after gene duplication: models, mechanisms, sequences, systems, and organisms. *J. Exp. Zool. Part B* 308B(1):58–73, 2007.
24. A. Stark, P. Kheradpour, S. Roy, and M. Kellis. Reliable prediction of regulator targets using 12 *Drosophila* genomes. *Genome Res.* 17:1919–1931, 2007
25. A. Tanay, A. Regev, and R. Shamir. Conservation and evolvability in regulatory networks: The evolution of ribosomal regulation in yeast. *Proc. Nat'l Acad. Sci. USA* 102(20):7203–7208, 2005.
26. S.A. Teichmann and M.M. Babu. Gene regulatory network growth by duplication. *Nature Genetics* 36(5):492–496, 2004.
27. R. Wang, Y. Wang, X. Zhang, and L. Chen. Inferring transcriptional regulatory networks from high-throughput data. *Bioinformatics* 23(22):3056–3064, 2007.
28. 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.
29. X. Zhang and B.M.E. Moret. Boosting the performance of inference algorithms for transcriptional regulatory networks using a phylogenetic approach. *Proc. 8th Workshop on Algs. in Bioinformatics WABI'08*, in *LNCS* **5251** (2008), pp. 245–258.