Network Inference of Dynamic Models by the Combination of Spanning Arborescences

Anthony Coutant, Céline Rouveirol

Network Inference of Dynamic Models by the Combination of Spanning Arborescences

Anthony COUTANT and Céline ROUVEIROL
Laboratoire d'Informatique de Paris Nord (LIPN), UMR CNRS 7030, Université Paris 13
99 avenue Jean-Baptiste Clément, 93430 Villetaneuse, France

Corresponding author: firstname.lastname@lipn.univ-paris13.fr

Abstract *In this paper, we tackle the problem of generative learning of dynamic models from "fat" time series data (high #variables/#individuals ratio), leading to a high sensitivity of learned models to the dataset noise. To overcome this problem, we propose a method computing a mixture of many highly biased but optimal spanning arborescences obtained from many perturbed versions of the original dataset, introducing variance to counterbalance the strong arborescence bias. The method is theoretically at the boundary between structure oriented Bayesian model averaging and recent work on density estimation using mixtures of poly-trees through a perturb and combine framework, transposed to a dynamic setting. In practice, preliminary results on the recent DREAM D8C1 challenge are promising.*

Keywords Network inference, Ensemble Learning, Model Averaging, Spanning Arborescences

1 Introduction

Accurate network inference, or generative structure learning, is a major problem in bioinformatics for the comprehension of systems of different kinds, such as regulatory networks [1] or cell signaling pathways [2]. Although a difficult problem in any dataset, generative structure learning on biological contexts is additionally challenged by the scarcity of the available datasets, obtaining such data often being very expensive and time consuming.

A major issue occuring in structure learning algorithms from data is the one of *data fragmentation*, where the computation of frequencies from data is not reliable enough to robustly estimate statistics and avoid overfitting. This problem occurs more particularly when the dataset is scarce and suffers from a very high variables/samples ratio, such as in many biological contexts where the number of available sets of measurements does not exceed several tens in a good situation, for a living organism made of hundreds or thousands of genes and involving many more molecules.

Biological systems are also often characterized by their dynamic properties, and many biological phenomenon under study are actually time evolving processes, e.g. the diauxic shift of the *Saccharomyces cerevisiae* yeast [3], for which available datasets can take the form of time series. Together, the data scarcity and the objective of finding dynamic models reinforce the data fragmentation issue, since considering several time points in space for each variable in the model both increases the number of actually considered variables in the network to learn, and reduces the samples size of the original dataset, due to the underlying used sliding window.

A possible strategy to prevent overfitting in such situation is to reduce the learning algorithm variance by reducing the number of possible models. This can be achieved for example by constraining the search space so that the resulting subspace have good properties, or by constraining the search algorithm so that a subset of possible models is reachable. Used alone, this strategy can find a good local, even global, optimum, but relatively to a potentally inadequate space where a good solution for the overall learning problem is unexistent.

Another possibility is to find an asymptotic structure which is the result of a consensus between different models learned from the dataset [4,5]. This way, the lack of sufficient statistics on data is partly offset by an attempt to compute sufficient statistics on potentially very noisy intermediate models.

In this article, we use both solutions. The general behaviour of the algorithm we propose is to compute an expected *composite* model using Bayesian Model Averaging [4] theory and a set of expected features, the expected edge existence, computed from the learning of many *component* models. These components are themselves highly biased models, more precisely spanning arborescences [6], with the interesting property for

these simple structure spaces, that a global optimum is computable in polynomial time for any dataset [7]. Due to this latter property, it is necessary to introduce variance in the dataset for each component learning task, in order to converge to diverse simple models, allowing different significant features of the final model to be represented in the population of simple models. This variance is obtained by: 1) perturbing the original dataset through sampling with replacement; 2) forbidding the use of some edges in the spanning arborescence, the edge blacklist being randomly generated for each component.

The paper is organized as follows. We first provide the background material through the description of required theory and previous works in section 2. Then, we describe our mixture algorithm in section 3. We finally validate in section 4 our algorithm with promising results on experiments over the challenging biological network inference application from time series data from the popular DREAM D8C1 recent challenge [8], before discussing the algorithm and its perspectives in section 5.

2 Related Works

Network inference has been studied for decades in bioinformatics context and many solutions can be grouped into the following families.

Feature selection methods break the global network inference problem into a set of N subproblems, N being the number of features in the studied biological system. The objective is to find the best neighbors of each feature in the biological model to learn, using various methods. A main strategy is to learn and rank, for each feature j, a set of (possibly linear) regression coefficients describing the function of all other features to j. Recent examples of this strategy include: the HLICORN method [9,10], where linear coefficients are computed between each possible gene and previously discovered coregulator sets; the GENIE3 [11] algorithm, where a supervised classification tree is built for each node, seen as the target, and potential parents are ranked according to their decreasing order of entropy gain between the tree's layers; BORUTA [12], a wrapper type of method around random forest classification; TIGRESS [13], which uses a *perturb and combine* strategy as this paper does. Note that this latter strategy is still significantly different from our proposal. First, TIGRESS reduces the learning problem into an independent learning problem for each node, while our proposal constrains the learned models globally, allowing here to enforce global sparsity and more generally allowing rich extensions to the method (e.g. global prior addition). Furthermore, the models to combine in this work are obtained by Lasso regression and forward selection strategy, without any quality guarantee, unlike our strategy. Finally, the approach only introduces variance in the process via data perturbation, while we propose both data and edge space sampling for each component, the latter showing significant experimental impact (cf. Section 4).

Ordinary Differential Equations (ODE) methods aim at modelling a dynamic biological system as a set of differential equations involving its features (genes, proteins, etc.). In practice, these methods can find highly accurate parameters for given equations, and have even been extended for learning the equations themselves [14], but their computational cost make them only suitable for small biological systems, which is not our target.

Pairwise score approaches aim at finding networks optimizing a sum of feature to feature scores measuring the level of correlation or dependency between them. A famous algorithm in this category is the ARACNE [15] method which computes a superset of the Chow-Liu spanning tree induced by the learning dataset, by first computing a complete graph with all pairwise mutual informations (MI), and then iteratively flagging edges for removal using a MI inequality for every triangle in the graph. Such posterior optimization, called *transitive reduction*, is indeed often very important in algorithms of this family since distinguishing between direct or indirect dependencies is not necessarily possible with the used measures.

(Dynamic) Bayesian network (DBN) learning algorithms [16] aim at finding the best factorization of the joint distribution involving the features at different consecutive time steps, as a product of conditional distributions (one per feature). Most DBN learning algorithms targetting problems of any size try to find a local optimum in the model space, through the use of a heuristic, going from one model in the space to another through iterative local perturbation of a best candidate obtained so far (best-first strategy) [17] or a set of best candidates (beam search) [18]. These algorithms often add some extra-mechanics to proceed further the first local optima, as for example random restarts or tabu search [19].

Most methods in the above mentioned families consider the input dataset as is to make network inference. In the context of small datasets, the data fragmentation issue together with the possibility of noise presence in

the dataset can misconduct the learning algorithms. In order to abstract from a single model learned from such dataset, ensemble learning strategies have been designed to implement a "wisdom of crowds" principle in a machine learning context. Historically used in a fully supervised context to average classification predictions and turn them into final majority decisions, these principles have been transposed for probability density estimation and structure learning contexts, a good example of such strategy for probabilistic models having been theorized under the Bayesian Model Averaging (BMA) [4] framework.

In BMA, the objective is to find an expected model, defined by a set of *expected features* it must satisfy, such as the dependency between its variables, the underlying graph edges or paths and so forth, assuming those features are relevant in the model space, by combining different *component* models. In practice, considering the whole space is intractable and one must rely on a subset to approximate the result, such as using Markov Chain Monte Carlo methods [20] or bagging [21].

Typical BMA methods consider the whole model space for learning which leads to the question of which heuristic to use and with which extra mechanics. This problem has been recently partially tackled in [5] but choices still remain. A recent work by Schnitzler et al. [22] considers combining more simple component models instead, namely spanning trees, and proposes extensions of the Chow-Liu algorithm [23] which show good results in their non-dynamic density estimation context. The algorithm proposed in this paper is close but differs in two main ways: it focuses at outputing an expected structure as in traditional BMA for structure learning, and thus require to make choices between the different component properties, while their work outputs a result in the form of a linear combination of each component's density probability function; it focuses on time series datasets instead of i.i.d. propositional data. Note that a preliminary adaptation of Schnitzler work for structure learning has been proposed in [24], but authors use the component models as an intermediate product to refine potential parent candidates of a greedy algorithm, while we propose an algorithm which directly infers a model from the components.

Among BMA methods, it is important to mention the seminal work in [4] about BMA structure learning through the computation of expected features using bagging on the Bayesian networks space. However, in this work, learned expected features are used as constraints for heuristic learning, which does not include component edges themselves, and whose result does not lead to significant improvement compared to the unconstrained learning counterpart. Finally, in this paper, while most work focuses on combining unconstrained Bayesian networks, few indications are also given on the results obtained by merging spanning trees, which apparently gave worse results than with unconstrained Bayesian networks. However, this can be explained by the low introduction of variance, unlike our proposal, which can lead to poor space exploration because of the components global optimality. Also, chosen experiment datasets can not be considered as being high dimensional ones and the variables/individuals ratio is less subject to data fragmentation, which can explain that more biased models perform worse. We will demonstrate in section 4 that combining arborescences in an adequate way can actually give very good results in this context.

3 Combining Spanning Arborescences for Network Inference

In this section, we propose a general algorithm for network inference from the combination of spanning arborescences. Several application oriented details are willingly not given, such as the specific scoring function used and the value of hyperparameters. These informations are detailed in Section 4 for the problem at hand.

3.1 Data representation

Let us consider a matrix representation of a dataset D consisting of n ordered time stamps (D rows) over N variables (D columns). Each column j is a sequence describing an observed variable over n time steps $\langle D_{ij} \rangle_{i \in \langle 1,\dots,n \rangle}$ and each row i describes the state of a system at time step i over the N observed variables. Our goal is to find a model of the system of interest in terms of dependencies between the observed variables at different time steps of the system. In this paper, we assume that this system is a Markov process, i.e. that each time step state only depends on the immediate previous step state, and that the transition from each state to the next state is driven by the same underlying model.

The first step for the proposed algorithm to work is to transform the $n \times N$ dataset into a $(n-1) \times 2N$ dataset D^t describing 2 consecutive time slices of the system. The transformation consists in concatenating

every pair of consecutive time steps from D into a "dynamic" example in D^t, i.e. ,we have for each row $D^t_i.$:

$$D^t_i. = [D_{i,1} \ \ldots \ D_{i,N} \ D_{i+1,1} \ \ldots \ D_{i+1,N}].$$

The given assumptions and the consequent representation allow transforming the ordered structure learning problem from D into a simpler i.i.d. structure learning problem in the dynamic examples space of D^t.

3.2 Learning component models

From a dynamic dataset D^t, the first learning step of the proposed algorithm is to compute a set of *component* models, i.e. simple models which will be combined in the second part of the algorithm.

Considering a number m of simple models to learn, we first compute m local perturbations of D^t, denoted by $\{D^{t[k]}\}_{1 \leq k \leq m}$, by sampling from D^t with replacement (bagging strategy [21]). Then, for each $D^{t[k]}$, a directed graph $G^k = (V, E^k)$, with V having one vertex for each of the N original variables, is built by first randomly choosing $\alpha \cdot N \cdot (N-1)/2$ undirected edges and then computing the two directed scores $s(A \to B)$ and $s(B \to A)$ for each of them. Finally, each graph G^k is searched for its optimum spanning arborescence A^k with respect to the score s, using the Edmonds algorithm [6].

Even if the built graphs only have one node per original variable (as opposed to two nodes, one for timestep t and one for timestep $t+1$), the semantics of an arc $X \to Y$ measures the influence of X at time t over Y at time $t+1$. This semantics is taken into account during scores' computation. Indeed, a score computation $s(X \to Y)$ is actually a score involving $D^{t[k]}_{.x}$ and $D^{t[k]}_{.(y+N)}$ columns of D^t, where x and y are the indices of variables X and Y in D. Concerning the choice of score itself, many asymmetrical scores can be used. A simple one with conditionals interpretation is the conditional entropy $H(X|Y)$. Bayesian scores, like Bayesian Dirichlet variants [7] can also be used with the advantage of being able to add prior information, relatively to each edge, to the learning task.

The edge sampling step before spanning arborescence computation is important in order to counterbalance the determinism of the Edmonds algorithm due to its global optimality. The choice of α at this step is critical and will be discussed in Section 4. On one hand, it must be low enough to avoid that the optimality of the spanning algorithm restricts the component models diversity too much before the combination step. On the other hand, it should also be high enough so that the spanning algorithm still discriminates between different models and the resulting components are not just the consequences of random sampling. The choice of sampling undirected edges instead of directly sampling directed ones is of major importance to this purpose, since it allows to obtain strongly connected G^k graphs, to guarantee the existence of a spanning arborescence for a wide range of α.

To conclude, the first learning step ensures each component to be sparse, and globally optimal in the considered arborescences space, with respect to s. Edmonds algorithm, like Kruskal one in the undirected graphs space, allows to get this optimality for each directed component in polynomial time.

3.3 Computing the composite model

Once the m component models have been learned, the second learning step aims at combining them into a composite model. In the Bayesian Model Averaging framework, this step is achieved by computing a set of *expected features* $\{\mathbf{E}(f_i)\}_i$ for the composite model, each $\mathbf{E}(f_i)$ being inferred from each component features set $\{f^k_i\}_{1 \leq k \leq m}$. In this paper, the feature space consists in the set of all possible edges in V^2, and an expected edge score is computed by counting how often that edge was present in the arborescence A^k, considering it was present in the initial weighted graph G^k. Formally, we have for all $(A, B) \in V^2$:

$$\mathbf{E}(f_{A \to B}) \approx \frac{|\,\{k \,|\, (A,B) \in \mathbf{edges}(A^k)\}\,|}{\alpha}.$$

More complex features could be considered, such as paths instead of edges or ancestor / descendant relationships, as in [4] (although the authors do not combine them in a single model). We leave these problems for future work since it would require more complex combination rules, requiring transitive reduction techniques [25], a difficult problem in the case where the input graph has cycles or is weighted.

The computation of all edges' expected scores in the composite model directly provides a ranking for those edges. A combined model is finally built from such ranking by choosing the k-best edges or all edges whose

score exceeds a given threshold. The ranking itself can be used to compare the learning results with an optimal model through an AUROC evaluation.

The Algorithm 1 summarizes the proposed approach.

Algorithm 1 The learning algorithm

Require:
D^t: a dynamic 2 slices of time learning dataset,
m: a number of component models to learn;
s: an edge directed weighting score;
α: a density for graphs setup pre-spanning arborescence;
σ: an edge weight threshold for final edges keep decision;
Ensure: a structural model from t to $t+1$ of the system
for $1 \leq k \leq m$ **do**
$D^{t[k]} := sample\_with\_replacement\_from(D^t)$
$G^k := build\_strongly\_connected\_graph(D^{t[k]}, s, \alpha)$
$A^k := edmonds\_spanning\_arborescence(G^k)$
$F^k := \mathbf{edges}(A^k)$
end for
$\forall (A,B) \in V^2 : \mathbf{E}(f_{A\to B}) := |\{k \mid (A,B) \in F^k\}| / \alpha$
return $G = choose\_top\_edges(\{\mathbf{E}(f_{A\to B})\}, \sigma)$

3.4 Complexity

Following the decomposition of the Algorithm 1, the time complexity can be expressed as the sum of two terms: one for the components computation, and another for the combination step. The components computation complexity is $m \cdot (s+g+e)$, where s (resp. g, e) is the complexity of sampling (resp. connected graph construction and Edmonds algorithm). The complexity of the sampling step is negligible here, but the construction of the graph G^k is in $\mathcal{O}(\alpha N(N-1)) \approx \mathcal{O}(N^2)$, as is the Edmonds algorithm computation with the Tarjan optimization for dense graphs [26] ($\mathcal{O}(N^2\, log\, N)$ for sparse ones). Thus the components computation part is in $\mathcal{O}(mN^2)$.

The combination part is trickier since it consists in a succession of joins between the component edgelists for further counting. Depending on the join algorithm used, this part can become the bottleneck of the overall learning approach. Indeed, a simple nested join has a time complexity in $\mathcal{O}(PQ)$ where P and Q are the number of rows in each table to join together. In our algorithm, this leads to a complexity in $\mathcal{O}(N^{2m})$. However, it is possible to considerably improve this step using better strategies, such as hash joins, running in $\mathcal{O}(P+Q)$, thus leading to a linear complexity in our settings. Note that in a purely sequential algorithm, it is not really necessary to compute joins since component edges can be counted just after arborescence computation. However, since this method is highly parallelizable due to the independence of components learning and of combinations order, it is preferable to consider this solution since the parallelization gain overcomes the joins cost in practice.

Overall, the proposed approach is the sum of a quadratic and a linear step (in a parallel configuration), and thus is of quadratic complexity.

4 DREAM 8 (HPN-DREAM) SC1B Network Inference Challenge Results

In this section, we validate our method and compare it with other algorithms through a dataset of the recent HPN-DREAM 8 Breast Cancer closed challenge [8].

4.1 Challenge and evaluation method description

The DREAM 8 SC1B subchallenge learning objective is to find the network of a synthetical biological model built using state of the art methods and biological knowledge. Simulation of this model led to the production of several time series involving 20 biological features. The data used to perform the validation of our algorithm was firstly pre-processed as described in section 3 to produce a single two-time slice dataset, the resulting data containing 80 t to $t+1$ examples over 40 temporal features.

The evaluation of learning results for this task is achieved by an official tool, the *DREAMTools* python package [27], through the computation of an AUROC score against the golden standard. In addition to com-

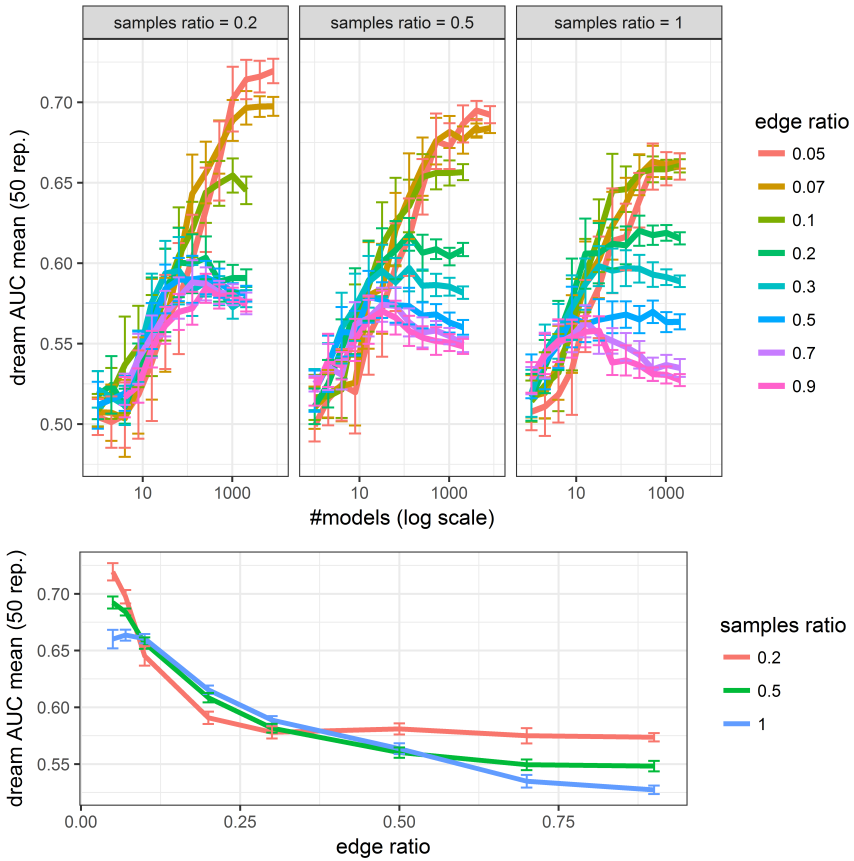

Fig. 1. (Top) mean and sds of AUC computed by DREAMTools over 50 computations as a function of the number of combined models, as well as the samples and edge ratio used for components learning. (Bottom) mean and sds of AUC computed by DREAMTools over 50 computations as a function of the edge and sample ratios. Only convergence values for increasing m are plotted.

puting scores the same way from one algorithm to another, this package also provides the expected ranking an algorithm would have reached if the challenge were still open, using all final results from the more than 100 official submissions, which allows for a cheap comparison with many algorithms of all families described in section 2.

In order to quantify the impact of several parameters on our algorithm learning quality, we have tested the method with different parametrization of the number of combined modes m, the ratio of samples n contained in each data perturbation, and the ratio of edges α present in each graph before each component learning. We used BDeu [7] gain as edge score, the difference between the $BDeu$ score of the $A \rightarrow B$ local structure and the no edge one. Namely for an edge $A \rightarrow B$: $BDeu(parents(B) = \{A\}) - BDeu(parents(B) = \emptyset)$.

4.2 Results

Results for many parametrizations, given in Figure 1(Top), show different clear trends. Firstly, we can see that for small edge ratios, the obtained AUC seems to monotonically increase with the number of combined models, until reaching plateaus. For bigger ratios, the trend seems to be mostly observable, but the higher the

sampling ratio, the lower the minimum edge ratio needs to be to show this trend. Additionally, we can observe that the convergence AUC value tends to increase whenever any of the edge or the sample ratio decrease, which is clearer in Figure 1(Bottom). These results seem to indicate that focusing on smaller parts of the available information for each component, while aggregating a higher number of them for final consensus, seem to give the best results, which confirms the requirement for components diversity in order to give a good consensus. Extra experiments done and not displayed here show that smaller edge and sample ratios break the observed trends. For edge ratio, this is predictable, since the minimum value displayed of 0.05 corresponds to an average number of 2 neighbors per node (considering we add the reverse edges for each sampled edge, to ensure we can compute a spanning arborescence), which is the minimal number of neighbors required for the algorithm to make a choice. Lower values actually lead the spanning arborescence algorithm to just select most available edges in the graphs it is given. For samples ratio, it seems to indicate a limit from which the dataset is too small to capture faithful enough information.

Concerning the expected ranking for the different results, our approach is very promising since it reached the 3^{rd} position for the best mean AUC obtained over the different parametrizations, outperforming GENIE3, ARACNE, all heuristic oriented Bayesian network methods, as well as all linear and most non-linear regression methods, all ODE and all ensemble learning solutions.

4.3 Discussion

At the moment, the gap with the best performance is of 0.045. A particularity of the considered DREAM subchallenge is that 3 out of the 20 biological features are actually fake nodes, supposed to have no correlation with the others. This shows a limitation of our approach in its current form: learning spanning trees means that every node will get one parent per component, even if there is no true correlation. Note that this problem is not necessarily easy to solve, since there is also a tendency for such spurious correlations to be non-uniformly distributed. Indeed, the optimization of the spanning arborescence score encourages to keep the apparently more correlated pairs of nodes, so the ones with the most biased noise are chosen. Since sample and edge samplings are uniformly done, there is a high probability for a restricted number of parents to appear in each component for a fake node. In practice, this means that a simple pruning of the components is not enough. Future work will address this issue.

5 Conclusion

In this paper, we have presented a network inference learning algorithm based on the combination of multiple spanning arborescences learned over multiple perturbation of the original dataset, with enforced diversity through edge sampling, showing promising results in practice on a recent DREAM challenge.

Experiments have more particularly shown that combining more models together with more diversity, involving a decrease of both sample and edge ratios in the currently defined parameter space, leads to better convergence values. It is encouraged to use this strategy in a quite extreme way, since best performances obtained in the experiments are achieved by situations where both ratios are very low. The only warning would be to still allow the Edmonds algorithm to have choice, in order not to make the components completely random. We have also seen in section 4 the impact of fake nodes on the results, and the difficulty of identifying them whenever the spanning arborescence assign most nodes a parent. This issue has a significant impact on the current results since removing edges involving fake nodes would lead to a top position of the approach.

Future works will address the limitations of the current algorithm, such as its sensitivity to fake nodes. More advanced extensions will also be investigated, such as the introduction of priors, really important in biological contexts, modular capabilities, which is becoming a standard in recent methods to abstract from a model complexity, and different component combination rules to preserve extra properties in the consensus model, such as paths or path lengths.

References

[1] Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene van Someren, and Reinhard Guthke. Gene regulatory network inference: Data integration in dynamic models—a review. *Biosystems*, 96(1):86 – 103, 2009.

[2] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[3] Ludwig Geistlinger, Gergely Csaba, Simon Dirmeier, Robert Küffner, and Ralf Zimmer. A comprehensive gene regulatory network for the diauxic shift in saccharomyces cerevisiae. *Nucleic acids research*, page gkt631, 2013.

[4] Nir Friedman, Moises Goldszmidt, and Abraham Wyner. Data analysis with bayesian networks: A bootstrap approach. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 196–205. Morgan Kaufmann Publishers Inc., 1999.

[5] Bradley M Broom, Kim-Anh Do, and Devika Subramanian. Model averaging strategies for structure learning in bayesian networks with limited data. *BMC bioinformatics*, 13(13):S10, 2012.

[6] Jack Edmonds. Optimum branchings. *Mathematics and the Decision Sciences, Part*, 1:335–345, 1968.

[7] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[8] Steven M Hill, Laura M Heiser, Thomas Cokelaer, Michael Unger, Nicole K Nesser, Daniel E Carlin, Yang Zhang, Artem Sokolov, Evan O Paull, Chris K Wong, et al. Inferring causal molecular networks: empirical assessment through a community-based effort. *Nature methods*, 13(4):310–318, 2016.

[9] I Chebil, Rémy Nicolle, G Santini, Céline Rouveirol, and Mohamed Elati. Hybrid method inference for the construction of cooperative regulatory network in human. *IEEE transactions on nanobioscience*, 13(2):97–103, 2014.

[10] Rémy Nicolle, François Radvanyi, and Mohamed Elati. Coregnet: reconstruction and integrated analysis of co-regulatory networks. *Bioinformatics*, page btv305, 2015.

[11] Va Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLOS ONE*, 5(9):1–10, 09 2010.

[12] Miron B Kursa and Witold R Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.

[13] Anne-Claire Haury, Fantine Mordelet, Paola Vera-Licona, and Jean-Philippe Vert. Tigress: Trustful inference of gene regulation using stability selection. *BMC Systems Biology*, 6(1):145, 2012.

[14] Alex Greenfield, Christoph Hafemeister, and Richard Bonneau. Robust data-driven incorporation of prior knowledge into the inference of dynamic regulatory networks. *Bioinformatics*, 29(8):1060–1067, 2013.

[15] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo D Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC bioinformatics*, 7(Suppl 1):S7, 2006.

[16] Min Zou and Suzanne D Conzen. A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.

[17] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[18] Peter Norvig. *Paradigms of artificial intelligence programming: case studies in Common LISP*. Morgan Kaufmann, 1992.

[19] Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[20] Walter R Gilks. Markov chain monte carlo. *Encyclopedia of Biostatistics*, 2005.

[21] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[22] François Schnitzler, Sourour Ammar, Philippe Leray, Pierre Geurts, and Louis Wehenkel. *Efficiently Approximating Markov Tree Bagging for High-Dimensional Density Estimation*, pages 113–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[23] C Chow and C Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[24] Sourour Ammar and Philippe Leray. *Mixture of Markov Trees for Bayesian Network Structure Learning with Small Datasets in High Dimensional Space*, pages 229–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[25] Andrea Pinna, Sandra Heise, Robert J Flassig, Alberto De La Fuente, and Steffen Klamt. Reconstruction of large-scale regulatory networks based on perturbation graphs and transitive reduction: improved methods and their evaluation. *BMC systems biology*, 7(1):73, 2013.

[26] Robert Endre Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[27] Thomas Cokelaer, Mukesh Bansal, Christopher Bare, Erhan Bilal, Brian M Bot, Elias Chaibub Neto, Federica Eduati, Alberto de la Fuente, Mehmet Gönen, Steven M Hill, et al. Dreamtools: a python package for scoring collaborative challenges. *F1000Research*, 4, 2015.