Combining tree-based and dynamical systems for the inference of gene regulatory networks

Citation for published version:

Digital Object Identifier (DOI):
10.1093/bioinformatics/btu863

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published in:
Bioinformatics

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Systems biology

Combining tree-based and dynamical systems for the inference of gene regulatory networks

Van Anh Huynh-Thu¹,* and Guido Sanguinetti¹,2,*

¹School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, and ²SynthSys - Systems and Synthetic Biology, University of Edinburgh, Edinburgh EH9 3JD, UK

*To whom correspondence should be addressed.

Abstract

Motivation: Reconstructing the topology of gene regulatory networks (GRNs) from time series of gene expression data remains an important open problem in computational systems biology. Existing GRN inference algorithms face one of two limitations: model-free methods are scalable but suffer from a lack of interpretability and cannot in general be used for out of sample predictions. On the other hand, model-based methods focus on identifying a dynamical model of the system. These are clearly interpretable and can be used for predictions; however, they rely on strong assumptions and are typically very demanding computationally.

Results: Here, we propose a new hybrid approach for GRN inference, called Jump3, exploiting time series of expression data. Jump3 is based on a formal on/off model of gene expression but uses a non-parametric procedure based on decision trees (called ‘jump trees’) to reconstruct the GRN topology, allowing the inference of networks of hundreds of genes. We show the good performance of Jump3 on in silico and synthetic networks and applied the approach to identify regulatory interactions activated in the presence of interferon gamma.

Availability and implementation: Our MATLAB implementation of Jump3 is available at http://homepages.inf.ed.ac.uk/vhuynht/software.html.

Contact: vhuynht@inf.ed.ac.uk or G.Sanguinetti@ed.ac.uk

Supplementary information: Supplementary data are available at Bioinformatics online.
the generative perspective enables principled predictions of expression levels under perturbations. However, model-based methods are not free from limitations: they tend to be computationally intensive, particularly in a Bayesian setting, and their parametric nature usually implies very stringent assumptions about the dynamics (e.g. linear), which may be difficult to justify biologically. Model-free methods avoid the pitfalls of model-based methods by greedily optimizing information-theoretic measures of co-variation between pairs of genes (Faith et al., 2007; Huynh-Thu et al., 2010; Margolin et al., 2006). Such methods typically have good scalability, enabling reconstructions of networks of hundreds of genes and have consistently achieved state-of-the-art reconstruction performance in comparative evaluations (Marbach et al., 2012). The lack of an underpinning model also enables great flexibility, as the interactions between genes are not constrained to follow a parametric functional representation. Such flexibility comes at a cost though: model-free methods, by their very nature, do not have clearly defined semantics in terms of dynamical systems and cannot be used for prediction in a straightforward way. Furthermore, incorporation of side information, which is natural in model-based methods, is generally challenging in model-free methods.

In this article, we aim to bridge the gap between model-based and model-free methods by proposing a hybrid approach to the network inference problem, called Jump3. Our approach starts from a well-defined, biologically plausible model of gene expression, the on/off model of gene expression (Ocone et al., 2013; Prashne and Gann, 2002), which we use to model the dynamics of individual nodes. Reconstruction of the edges is instead based on a non-parametric, tree-based method modelled on the state-of-the-art GENIE3 method (Huynh-Thu et al., 2010). Adapting the tree-based method to the probabilistic setting is a novel challenge in machine learning and involves devising a novel decision function for learning the tree. Here, we introduce the ‘jump tree’, which uses the marginal likelihood of the node’s dynamical model as a decision function. This choice has several benefits: it embeds the tree-based learning procedure in the probabilistic method, effectively grounding it as a greedy solution to structure learning in a large latent-variable model. Furthermore, the use of the marginal likelihood means our method inherits the ease with which side information can be incorporated in probabilistic models. Our experiments with both synthetic and real data show that Jump3 has good scalability and achieve competitive or better results than state-of-the-art alternatives.

2 Model and methods

Here, we describe Jump3, a hybrid approach for GRN inference that is based on a formal dynamical model of the expression of each gene of the GRN and that employs a greedy, non-parametric, method for reconstructing the topology of the GRN. Exploiting time series of expression data, Jump3 assigns a confidence score to each putative regulatory link of the GRN. Note that in this article, we leave open the problem of choosing a threshold on the weights to obtain a practical network and focus on providing a ranking of the regulatory links.

2.1 Gene expression model

At the heart of our framework, we use the on/off model of gene expression (Prashne and Gann, 2002), a simple, yet plausible, model where the rate of transcription of a gene can vary between two levels depending on the activity state \( \mu \) of the promoter of the gene.

\[
\frac{dx_i(t)}{dt} = (A_i \mu_i(t) + b_i - \lambda_i x_i)dt + \sigma dw(t),
\]

where subscript \( i \) refers to the \( i \)-th target gene. Here, the promoter state \( \mu_i(t) \) is a binary variable (the promoter is either active or inactive), which depends on the expression levels of the transcription factors (TFs) that bind to the promoter (see Fig. 1). \( \Theta_i = \{ A_i, b_i, \lambda_i \} \) is the set of kinetic parameters. \( A \) represents the efficiency of the promoter in recruiting polymerase when being in the active state, \( b_i \) denotes the basal transcription rate and \( \lambda_i \) is the exponential decay constant of \( x_i \). The term \( \sigma dw(t) \) represents a white noise-driving process with variance \( \sigma^2 \).

For a given trajectory of the promoter state, i.e. when we are given the states \( \mu_i(t), \forall t \), the SDE (1) is linear and its solution \( x_i(t) \) is equivalent to a Gaussian Markov process, i.e. an Ornstein–Uhlenbeck (OU) process (Gardiner, 1996). The mean \( m_i(t) \) and covariance \( c_i(t, t') \) functions of this OU process are given by:

\[
m_i(t) = x_i(0) e^{-\lambda t} + \int_0^t e^{-(\lambda - \delta)s} \mu_i(s)ds + \frac{b_i}{\lambda_i}(1 - e^{-\lambda t}),
\]

\[
c_i(t, t') = \frac{1}{\lambda_i} (e^{-\lambda(t-t')} - e^{-\lambda(t-t')}).
\]

Note that the covariance function contains two terms, one that is stationary \( (e^{-\lambda(t-t')}) \) and one that is non-stationary \( (e^{-\lambda(t-t')}). \) The second term is typically much smaller than the first one and thus could be neglected in practice. We, however, assume that a perturbation is applied to the network at \( t = 0 \), and we use the covariance function with its non-stationary term to take into account the initial transient behaviour of the network.

Let us assume that the gene expression \( x_i \) is observed with i.i.d. Gaussian noise at a finite number \( N \) of time points:

\[
\hat{x}_{i,k} = x_i(t_k) + \epsilon_{i,k}, \quad \epsilon_{i,k} \sim \mathcal{N}(0, \sigma^2_{i,k}), k = 1, \ldots, N,
\]

where \( \sigma^2_{i,k} \) is the variance of the observation noise at time point \( t_k \). As a consequence, the observed expression levels follow a multivariate normal distribution:

\[
x_i \sim \mathcal{N}(m_i, C_i + D_i),
\]

where \( m_i = [m_i(t_1), m_i(t_2), \ldots, m_i(t_N)]^T \), \( C_i \in \mathbb{R}^{N \times N} \) denotes the covariance matrix, with \( C_i[k, l] = c_i(t_k, t_l) \) and \( D_i \in \mathbb{R}^{N \times N} \) is a diagonal matrix with the values \( \sigma^2_{i,k} \) along the diagonal. One can

![Fig. 1. Example of GRN. Circles represent the observed gene expressions, and squares represent the latent promoter states. Thick arrows model the promoter activations and show the network topology](http://bioinformatics.oxfordjournals.org/Downloaded from at Edinburgh University on June 22, 2015)
therefore compute the marginal log likelihood of the observations, given by:

\[
\mathcal{L} = \log p(\mathbf{x}_i) = -\frac{N}{2} \log(2\pi) - \frac{1}{2} \log(|C_i + D_i|) \\
- \frac{1}{2} (\mathbf{x}_i - \mathbf{m}_i)^\top (C_i + D_i)^{-1} (\mathbf{x}_i - \mathbf{m}_i).
\] (2)

Notice that this probabilistic formulation allows for a natural incorporation of replicate information by simply multiplying the likelihoods of replicate profiles.

Within this context, our goal is, for each target gene \(i\):

1. To identify the promoter state trajectory \(\mu_i\) over the time interval \([0, t_N]\) that maximizes the log likelihood \(\mathcal{L}\);
2. To identify the regulators of the target gene, i.e. the genes whose expression levels influence \(\mu_i\).

Both problems are jointly addressed by using a non-parametric approach described in the next section and illustrated in Figure 2.

### 2.2 Network reconstruction with jump trees

In our model, we make the assumption that the state of the promoter of a target gene \(i\) is a function of the expression levels of the genes that are direct regulators of gene \(i\), i.e. the genes that are directly connected to gene \(i\) in the targeted network (Fig. 1). Denoting by \(\mathbf{x}_{\text{reg}, i}(t)\) the vector containing the expression levels at time \(t\) of the regulators of gene \(i\), we can write:

\[
\mu_i(t) = f_i(\mathbf{x}_{\text{reg}, i}(t)) + \zeta_i, \forall t,
\]

where \(\zeta_i\) is a random noise with zero mean. Recovering the regulatory links pointing to gene \(i\) thus amounts to finding the genes whose expression is predictive of the promoter state \(\mu_i\). To achieve this goal, we propose a procedure based on decision trees, which computes confidence scores \(w_{ij}\), \(\forall j \neq i\), measuring the importance of each gene \(j\) in the prediction of the state \(\mu_i\).

#### 2.2.1 Decision trees with a latent output variable

Tree-based methods have been applied successfully in the inference of GRNs (Huynh-Thu et al., 2010) and have appealing properties (Geurts et al., 2009). First, they are non-parametric and hence do not make any assumption about the nature of the function \(f_i\), which can be non-linear. Another advantage of tree-based methods is their ability to detect multivariate interacting effects between features. This is a non-negligible advantage when inferring GRNs, since the regulation of gene expression is expected to be combinatorial, i.e. to involve several regulators. Tree-based methods are also essentially parameter-free, and since their computational complexity is at most linear in the number of features, they can deal with high-dimensionality.

The basic idea of our GRN inference procedure is to learn for each target gene \(i\) a model \(f_i\) in the form of a decision tree (or an ensemble of decision trees), which predicts the promoter state \(\mu_i\) at any time \(t\) from the expression levels of the candidate regulators at the same time \(t\). However, standard tree-based methods cannot be applied here since the output \(\mu_i(t)\) is a latent variable. We therefore propose a new decision tree algorithm called ‘jump tree’. (In stochastic process theory, the discrete variable \(\mu_i(t)\) is called a jump process. The term ‘jump tree’ thus refers to a tree that predicts such a jump process.) Briefly, a jump tree is constructed top-down using a greedy algorithm and partitions the set of observation time points into different subsets based on tests on the expression levels of the candidate regulators. Each terminal node (or leaf) of the jump tree then corresponds to a subset of time points at which \(\mu_i\) is either 0 or 1. While in a standard decision tree the observations are split based on the minimization of the entropy of the output variable, in a jump tree the split is performed based on the maximization of the likelihood of the observations \(\mathbf{x}_i\).
More formally, the different steps for learning a jump tree predicting the latent variable $\mu_i$ are the following:

1. **Initialization.** Start with the simplest tree, which is only composed of one leaf. This leaf contains the whole set of $N$ observation time points, and $\mu_i(t) = 0$, ∀$t$, with a corresponding log likelihood $\mathcal{L}$.

2. **Creation of a split node.** Each iteration of the greedy algorithm consists in creating a split node from a leaf $\mathcal{N}$ and updating the promoter state trajectory and the likelihood. Given the current jump tree, the current promoter state trajectory $\mu_i$ and the current log likelihood $\mathcal{L}$ (obtained after the previous iteration), the set $\mathcal{N}_k$ of observation time points of the leaf $\mathcal{N}$ is partitioned using the following procedure:
   a. **Definition of a split.** Given the observed expression $x_i$ of a candidate regulator $j \neq i$ and a threshold value $c$, a candidate promoter state trajectory $\mu_i^{c,j}$ is obtained by setting:
      $\mu_i^{c,j}(t_k) = \begin{cases} 
0, & \text{if } \hat{x}_i(t_k) < c, \\
1, & \text{if } \hat{x}_i(t_k) \geq c,
\end{cases}$
   for each time point $t_k \in \mathcal{N}_k$. For the time points that do not belong to $\mathcal{N}_k$, the promoter states are kept the same:
      $\mu_i^{c,j}(t_k) = \mu_i(t_k), \forall t_k \notin \mathcal{N}_k$.
   Between two observation time points $t_k$ and $t_{k+1}$, the states $\mu_i^{c,j}(t), t_k < t < t_{k+1}$, are merely set to the state obtained at time point $t_k$. Note that the condition $j \neq i$ can be relaxed to incorporate autoregulation; however, in our experiments we have kept it to improve identifiability.
   b. **Evaluation of the split.** The best candidate regulator $j_i$ and threshold $c_i$ are selected, i.e. those ones that yield the maximum likelihood:
      $\mathcal{L}_i = \max_{j, c} \mathcal{L}(\mu_i^{c,j}),$
      $(j_i, c_i) = \arg \max_{j, c} \mathcal{L}(\mu_i^{c,j}),$
   where $\mathcal{L}(\mu_i^{c,j})$ is the likelihood obtained with the trajectory $\mu_i^{c,j}$.
   c. **Decision and update.** If the likelihood is increased, i.e. $\mathcal{L}_i > \mathcal{L}$, then:
      - Replace the leaf $\mathcal{N}$ with a split node containing the optimal test ‘$x_i < c_i$’;
      - Split $\mathcal{N}_k$ into two subsets $\mathcal{N}_0$ and $\mathcal{N}_1$ according to this test;
      - The child nodes of the new split node are two leaves, containing, respectively, $\mathcal{N}_0$ and $\mathcal{N}_1$;
      - Update the promoter state trajectory: $\mu_i \leftarrow \mu_i^{c,j_i}$;
      - Update the log likelihood: $\mathcal{L} \leftarrow \mathcal{L}_i$.

3. **Selection of the leaf.** The order in which the leaves are turned into split nodes change the final value of the likelihood $\mathcal{L}$. In our procedure, the jump tree is grown using a best-first strategy, i.e. at each iteration, steps 2a and 2b are repeated for each leaf of the current tree and the leaf that yields the highest maximum likelihood $\mathcal{L}_i$ is selected. Step 2c is then applied to this leaf. This procedure is illustrated in Figure 3.

4. **Stop.** The algorithm stops when $\mathcal{L}$ cannot be increased anymore, i.e. when $\mathcal{L} \leq \mathcal{L}$ for each leaf of the current tree. The algorithm then outputs the current jump tree and the current trajectory of the promoter state $\mu_i$.

The jump tree pseudo-code can be found in Section 1 of the Supplementary Information.

### 2.2.2 Ensemble of decision trees

A fully grown decision tree typically overfits the observed data, and significant improvements can be obtained with ensemble methods that average the predictions of several randomized trees, e.g. Random Forests (Breiman, 2001) or Extra-Trees (Geurts et al., 2006).

In Jump3, we use the Extra-Trees procedure, which randomizes the test at each split node of a tree (in step 2 of the jump tree algorithm). Rather than testing all the possible combinations of candidate regulator $j$ and threshold $c$, the best split is determined among $K$ random splits, each obtained by randomly selecting one candidate regulator (without replacement) and a threshold value. The prediction of $\mu_i(t)$ is then averaged over the different trees of the ensemble, yielding a probability for the promoter state to be active at time $t$.

### 2.2.3 Importance measure

The learned tree-based model is used to derive an importance score for each candidate regulator, quantifying the relevance of that candidate regulator for the prediction of $\mu_i(t)$. The importance $w_{ji}$ of a candidate regulator $j_i$ is then used as weight for the putative regulatory link of the network that is directed from gene $j$ to gene $i$.

We propose a measure that, at each split node $\mathcal{N}$, computes the increase of the likelihood due to the split:

$$I(\mathcal{N}) = \mathcal{L}(\mu_i^{c,j_i}) - \mathcal{L}(\mu_i)$$

where $\mathcal{L}(\mu_i)$ and $\mathcal{L}(\mu_i^{c,j_i})$ are the log likelihoods, respectively, obtained before and after the split on $\mathcal{N}$. For a single tree, the overall importance $w_{ji}$ of one candidate regulator $j_i$ is then computed by summing the $I$ values of all tree nodes where this regulator is used to split:

$$w_{ji} = \sum_{k=1}^{n} I(\mathcal{N}_k) g(\mathcal{N}_k, j_i)$$

where $n$ is the number of split nodes in the tree and $\mathcal{N}_k$ denotes the $k$th split node. $g(\mathcal{N}_k, j_i)$ is function that is equal to one if the candidate regulator $j$ is the one selected at node $\mathcal{N}_k$ and zero otherwise. The candidate regulators that are not selected at all thus obtain an importance score of zero and those ones that are selected close to the root node of the tree typically obtain high scores. Importance measures can be easily extended to ensembles of trees, by simply averaging the importance scores over all the trees of the ensemble.

### 2.2.4 Regulatory link ranking

Each tree-based model $f_i$ yields a separate ranking of the genes as potential regulators of a target gene $i$ in the form of importance scores $w_{ji}$. For a single tree, the sum of the importance scores of all candidate regulators is equal to the total increase of likelihood yielded by the tree:

$$\sum_{j\neq i} w_{ji} = \mathcal{L}(\mu_i) - \mathcal{L}(0),$$

where $\mathcal{L}(0)$ is the initial log likelihood obtained with $\mu_i(t) = 0$, ∀$t$ and $\mathcal{L}(\mu_i)$ is the final log likelihood obtained after the tree has been grown. As a consequence, if we trivially order the regulatory links according to the scores $w_{ji}$, this is likely to introduce a positive bias for the regulatory links that are directed towards the genes for which the overall likelihood increase is high. To avoid this bias, we normalize the importance scores obtained from each tree, so that they sum up to one:

$$w_{ji} \leftarrow \frac{w_{ji}}{\mathcal{L}(\mu_i) - \mathcal{L}(0)}$$
2.3 Computational complexity

Since the value of the parameter \(k\) is not optimized (see the details in the Supplementary Information), the computation of the covariance matrix \(C_i\) and the inversion of the matrix \(C_i + D_i\), which are required for the computation of the log likelihood \(L\), are done only once for each target gene. Therefore, the runtime complexity of Jump3 comes mainly from the optimization of the parameters \(A_i\) and \(b_i\) and the matrix multiplication in the last term of Equation (2), which are iteratively repeated during tree growing. Both parameter optimization and matrix multiplication have a complexity that is on the order of \(O(N^2)\), where \(N\) is the number of observations. Let us assume for simplicity that each tree that is learned contains \(S\) splits. It can be shown that the complexity for growing an ensemble of jump trees using the Extra-Trees procedure is \(O(TKS^2N^2)\), where \(T\) is the number of trees and \(K\) is the number of randomly chosen candidate regulators when searching for the optimal split at a node. The complexity of Jump3 is thus \(O(pTKS^2N^2)\) since it requires to build an ensemble of trees for each of the \(p\) genes of the network. At worst, the complexity of the algorithm is thus quadratic with respect to the number of genes (when \(K = p - 1\)) and \(O(N^3)\) with respect to the number of observations (when \(S = N - 1\), i.e. each tree is fully developed with each leaf corresponding to one time point). However, this worst case scenario never happens in practice; \(S\) is usually much lower than \(N\).

Table 1 gives an idea of the computing times, using our MATLAB implementation with \(K\) set to the number of candidate regulators and 100 trees per ensemble. These computing times were measured on an 8-GB RAM, 1.7 GHz Intel core i7 computer. Note that the large amount of time required to infer a DREAM4 size-100 network is due to the high number of observations. Such a high number is usually not encountered in real datasets, where the number of observations is typically much lower than the number of genes.

The Jump3 algorithm can be easily parallelized over the \(p\) genes, as well as over the different trees of an ensemble.

2.4 Performance metrics

Jump3 provides a ranking of the regulatory links from the most confident to the least confident. To evaluate such a ranking independently of the choice of a specific threshold, we use the precision-recall (PR) curve and the area under this curve (AUPR). The PR curve plots, for different thresholds on the weights of the links, the proportion of true positives among all predictions (precision) versus the percentage of true positives that are retrieved (recall). A perfect ranking, i.e. a ranking where all the positives are located at the top
of the list, yields an AUPR equal to one, while a random ranking results in an AUPR close to the proportion of positives (i.e. close to zero since the proportion of true links among all possible links in a network is usually very low).

### 3 Results

We evaluated the proposed Jump3 procedure on several in silico networks as well as one synthetic network (IRMA). As a case study, we applied the procedure to expression data from macrophages treated with interferon gamma (IFN$_\gamma$), to identify IFN$_\gamma$:activated regulatory interactions. In all our experiments, ensembles of 100 trees were grown and the main parameter $K$ of the Extra-Trees was set to the number of input candidate regulators. For the in silico and IRMA networks, $K = p - 1$, where $p$ is the number of genes in the network and $K = 40$ in the case of the IFN$_\gamma$ network (see later for the description of that experiment).

#### 3.1 In silico networks

We evaluated Jump3 on the networks of the DREAM4 In Silico Network challenge (Marbach et al., 2012; Prill et al., 2010), which are 5 networks of 10 genes and 5 networks of 100 genes. For each network topology, two types of simulated expression data were used:

- **Toy data**: we simulated the expression data using the on/off model based on Equation (1). A network perturbation was simulated through a switch in the promoter state of some genes and given a set of parameters $\Theta_i = \{A_i, b_i, \lambda_i\}$ for each gene $i$, the model was simulated to produce continuous time series for both promoter states and gene expressions. Noisy observations at discrete time points were obtained from the expression time series by adding i.i.d. Gaussian noise. The toy data are available in the Supplementary Material.

- **DREAM4 data**: we applied Jump3 to the time series data that was provided in the context of the DREAM4 challenge. Each time series experiment consisted in strongly increasing or decreasing the initial expression of about one-third of the genes, thereby simulating a physical or chemical perturbation. The perturbation was applied to the network at time $t = 0$ and was removed after 10 time points, making the system return to its original state. For each network of 10 (respectively 100) genes and each simulation type, noisy observations were sampled at 21 time points under 5 (respectively 10) different network perturbations, for a total of 105 (respectively 210) observations per gene.

First, we checked the quality of the data modelling that is obtained with Jump3. Results on the toy and DREAM4 data are, respectively, shown in Figure 4 and Supplementary Figure S1 (in the Supplementary Material), for one gene of a size-100 network. We notice from a qualitative point of view that Jump3 returns a good prediction of the promoter state and that the on/off model has sufficient flexibility to provide a good fit of the gene expression, as shown before (Ocone et al., 2013; Opper et al., 2010).

Next, we evaluated the performance of the method in terms of network reconstruction and we compared it to other existing network inference procedures: two model-free methods, which are time-lagged variants of GENIE3 (Huynh-Thu, 2012) and CLR (Faith et al., 2007), respectively; two model-based methods, namely Inferelator (Greenfield et al., 2010) and TSNI (Bansal et al., 2006), and G1DBN (Lébre, 2009), a method based on dynamic Bayesian networks. For TSNI, a separate network was inferred for each perturbation, and a consensus network was computed as the average of the different inferred networks. For all the remaining methods, networks were inferred using the complete dataset (all perturbations simultaneously). GENIE3 was applied with the Extra-Trees, the parameter $K$ set to the number of candidate regulators, and ensembles of 100 trees. TSNI was used with two principal components. The other methods were run using the default values of the parameters.

AUPR values obtained for the size-100 networks are shown in Tables 2 and 3, for the toy and DREAM4 data, respectively. Results on the size-10 networks are shown in Supplementary Table S2. In the case of the toy data, Jump3 yields the highest AUPR for each network. As expected, its performance decreases when the networks are inferred from the DREAM4 data, due to the mismatch between the on/off model and the one used to simulate the data. For the small networks of 10 genes, CLR, Inferelator and G1DBN have the best performances, without a clear winner. Jump3 seems robust when inferring large networks, since it outperforms the other methods on the size-100 networks. Note that the official best methods of the DREAM4 challenge obtained higher AUPR levels because they used additional interventional (knockout and knockdown) data.

#### 3.2 The synthetic IRMA network

The different GRN inference methods were applied to reconstruct the IRMA (In vivo Reverse-engineering and Modeling Assessment) network, a synthetic GRN embedded in the budding yeast Saccharomyces cerevisiae (Cantone et al., 2009). This network is composed of 5 genes and 6 regulatory interactions and can be activated and deactivated in the presence of galactose and glucose, respectively. The expression levels of the five genes were measured using quantitative RT-PCR during the transition from glucose to galactose ('switch-on' time series of 16 time points), as well as during the transition from galactose to glucose ('switch-off' time series of 21 time points).

As shown in Table 4, Jump3 is competitive with the two model-based methods (Inferelator and TSNI) when inferring the network from the switch-on data. In the case of the switch-off data, Jump3 yields the best performance. Notice that while the model-free methods (GENIE3 and CLR) typically perform better than the model-based methods on the in silico networks, the opposite is observed here on the IRMA network. This shows that model-based methods can be very powerful on very small networks, but their performances rapidly degrade as the number of genes in the network increases.

Promoter state predictions and gene expression fits obtained with Jump3 are shown in Supplementary Figures S2 and S3.

#### 3.3 The IFN$_\gamma$ network

Finally, we applied Jump3 to gene expression data from murine bone marrow-derived macrophages (Blanc et al., 2011). The macrophages were treated with interferon gamma (IFN$_\gamma$) and gene expression levels were measured at 25 half-hourly time points over 12 h, using Agilent microarray platform. We focused our analysis on the 1000 genes whose expression vary the most across the time series. Forty of these genes were classified as TFs by Gray et al. (2004), and we applied Jump3, GENIE3 and CLR to identify regulatory interactions between these 40 TFs and all the 1000 genes.

The 500 top-ranked regulatory links predicted by each method are shown in Figure 5A and supplementary Figure S4. (Cytoscape files for these three predicted IFN$_\gamma$ networks are also available in the
Supplementary Material.) As can be seen in these figures, the predicted networks are highly modular with a few TFs acting as hubs and regulating a large number of target genes (although the modules of the CLR networks are less distinct). Figure 5B shows the (empirical) node degree distribution of the Jump3 network. Although the networks of GENIE3 and CLR share a relatively large number of edges, Jump3 yields very different predictions (Fig. 5C), indicating that the addition of a dynamical model significantly alters the networks found.

Several of the hub TFs (defined as TFs predicted as having >10 targets and listed in Table 5) have biologically relevant annotations: apart from the interferon responsive TFs Irf1 and Irf7, we find Hoxc6 (associated with cytomegalovirus infection) and cancer-associated TFs such as Egr1, Bmyc and Pbx2, reinforcing the deep connections of the immune response with cancer (de Visser et al., 2006). Quantitative evaluations of these results in terms of enrichment for known regulatory links are hampered by the absence of large-scale gold standards for human regulatory networks. The widely used TRANSFAC database (http://www.gene-regulation.com/pub/databases.html) only reports information for a handful of TFs included in this analysis, and the number of known targets among the selected 1000 genes is usually very low (one or two at maximum), precluding a systematic enrichment analysis. The human homologues of three hub TFs (Egr1, Bmyc and Irf1) were assayed using ChIP-Seq by the ENCODE consortium (The ENCODE Project Consortium, 2012), providing a potentially much larger number of putative targets. An analysis of this data is reported in the Section 2 of the Supplementary Material and shows considerably higher recall for Jump3 (compared with GENIE3 and CLR) and a higher precision for two of the three TFs. Nevertheless, these numbers (only three TFs) are still very small for an enrichment analysis, which is in any case weakened by the data coming from a different organism in different experimental conditions.

4 Discussion

Elucidating the topology of GRNs is a fundamental step towards our understanding of how a cell or an organism can respond to its environment. Despite years of concerted efforts by the computational biology community, this task is still far from complete and a unified framework for GRN inference remains elusive. Here, we presented Jump3, a novel approach to GRN inference, which attempts to combine the interpretability of model-based methods with the scalability of greedy, model-free methods, thus bridging the gap between the two main classes of GRN inference approaches.

Experiments on simulated and synthetic data show that Jump3 is always competitive and often outperforms state-of-the-art GRN inference procedures, while an experiment on a real dataset shows its potential for biologically meaningful hypothesis generation. The on/off model is flexible enough to allow good fits of the data.

While we believe that Jump3 is a step in the right direction, we also acknowledge that the complexity of gene regulation will pose a strict limit to the potential of GRN inference from expression data.
alone. A first limitation comes from the assumption that the messenger RNA level can be used as a proxy for the protein activity, which is often not correct (Vogel and Marcotte, 2012). A simple improvement of Jump3 would thus be the exploitation of protein data, which are becoming less rare gradually, to predict the target promoter states. Another important direction is the integration in GRN inference algorithms of complementary data, such as microRNA expression, chromatin, protein-protein interactions or microbiomes and some promising initial steps in this direction are being taken (e.g. Ellwanger et al., 2014; Greenfield et al., 2013). The probabilistic generative model underlying Jump3 would allow the incorporation of additional information in a natural way via a modification of the likelihood function, while the non-parametric tree-based approach would ensure the scalability of the whole procedure.

Using the method on large networks with relatively few observations may incur co-linearity problems, i.e. genes that have very similar profiles leading to confounding factors in the inference. A simple fix to this would be to pre-process the data with some clustering algorithm; this would further increase scalability, at the cost of some interpretabiltiy.

Ultimately, a major limitation for many studies in computational biology is the lack of systematic, large-scale gold standards on which to evaluate the models; this generalized fact reinforces the need for a tight coupling between experimental and theoretical research, and we hope that inference methods such as Jump3 could be useful in prioritizing experimental designs.

**Acknowledgement**

We thank Peter Ghazal and Thorsten Forster for useful discussions on interferon gamma biology.

**Funding**

This work was supported by the European Research Council [grant number MSC306999].

**Conflict of Interest** none declared.

**References**


---

**Table 5. Hubs of the predicted IFN$\gamma$ networks**

<table>
<thead>
<tr>
<th></th>
<th>Jump3</th>
<th>GENIE3-lag</th>
<th>CLR-lag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bmyc</td>
<td>(31)</td>
<td>(18)</td>
<td>(14)</td>
</tr>
<tr>
<td>Egr1</td>
<td>(70)</td>
<td>(25)</td>
<td>(15)</td>
</tr>
<tr>
<td>Egr4</td>
<td>(51)</td>
<td>(53)</td>
<td>(41)</td>
</tr>
<tr>
<td>Hexc6</td>
<td>(89)</td>
<td>(139)</td>
<td>(46)</td>
</tr>
<tr>
<td>Irf1</td>
<td>(30)</td>
<td>(18)</td>
<td>(30)</td>
</tr>
<tr>
<td>Irf7</td>
<td>(12)</td>
<td>(18)</td>
<td>(58)</td>
</tr>
<tr>
<td>Mrg2</td>
<td>(22)</td>
<td>(18)</td>
<td>(34)</td>
</tr>
<tr>
<td>Myod1</td>
<td>(21)</td>
<td>(27)</td>
<td>(58)</td>
</tr>
<tr>
<td>Pbx2</td>
<td>(13)</td>
<td>(27)</td>
<td>(32)</td>
</tr>
<tr>
<td>Pou3f1</td>
<td>(38)</td>
<td>(27)</td>
<td>(32)</td>
</tr>
<tr>
<td>Stat2</td>
<td>(20)</td>
<td>(30)</td>
<td>(20)</td>
</tr>
<tr>
<td>Sox13</td>
<td>(58)</td>
<td>(11)</td>
<td>(11)</td>
</tr>
</tbody>
</table>

For each TF, the number of predicted targets is indicated between parenthesis.

**Fig. 5.** (A) IFN$\gamma$ network predicted by Jump3. The network was drawn using Cytoscape (Wang et al., 2012). (B) Node degree distribution of the network in (A). (C) Number of shared edges between the networks predicted by Jump3, GENIE3 and CLR


