Network Reconstruction

Maximum Likelihood Reconstruction of Ancestral Networks by Integer Linear Programming

Vaibhav Rajan [1,*], Carl Kingsford [2], Xiuwei Zhang [3,4]

[1] Department of Information Systems and Analytics, School of Computing, National University of Singapore, Singapore.
[2] Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.
[3] Deparment of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA.
[4] Ragon Institute of Massachusetts General Hospital, MIT and Harvard, Cambridge, MA, USA

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: The evolutionary history of biological networks enables deep functional and evolutionary understanding of various bio-molecular processes. Network growth models, such as the Duplication-Mutation with Complementarity (DMC) model, provide a principled approach to characterizing the evolution of protein-protein interactions (PPI) based on duplication and divergence. Current methods for model-based ancestral network reconstruction, primarily use greedy heuristics and yield sub-optimal solutions.
Results: We present a new Integer Linear Programming (ILP) solution for maximum likelihood reconstruction of ancestral PPI networks using the DMC model. By construction, our model is designed to find the optimal solution. It can also use efficient heuristics from general-purpose ILP solvers to obtain multiple optimal and near-optimal solutions that may be useful in many applications. Experiments on synthetic and real data show that our ILP obtains solutions with higher likelihood than those from previous methods. We evaluate our algorithm on two real PPI networks, with proteins from the families of bZIP transcription factors and Commander complex. On both the networks, solutions from our ILP has higher likelihood and are in better agreement with independent biological evidence from other studies.
Availability: A Python implementation is available at https://bitbucket.org/cdal/.
Contact: vaibhav.rajan@nus.edu.sg

1 Introduction

An organism's genotype and phenotype is mediated by complex biological interactions. Snapshots of such interactions are graphically captured by networks and spatio-temporal analysis of biological networks has led to deep functional and evolutionary understanding of molecular and cellular processes (Yamada and Bork, 2009). Knowledge of the evolution of networks such as Protein-Protein Interactions (PPI), metabolic and gene regulatory networks has been effectively used in the study of: molecular mechanisms in yeast (Wagner, 2001), cell signaling and adhesion genes (Nichols *et al.*, 2006), modularity in metabolic networks of bacterial species (Kreimer *et al.*, 2008), and of protein complexes (Pereira-Leal *et al.*, 2006), functional modules from conserved ancestral protein-protein interactions (Dutkowski and Tiuryn, 2007), evolutionary trends of biosynthetic capacity loss in parasites (Borenstein and Feldman, 2009), regulatory network inference (Zhang and Moret, 2010) and essential and disease-related genes in humans (Vidal *et al.*, 2011).

Generative models, called network growth models, that describe the evolution of networks have been used to explain properties of networks in other domains, such as the Preferential Attachment Model (Barabási and Albert, 1999) (for the World Wide Web) and the Forest Fire Model (Leskovec *et al.*, 2005) (for social networks). These models encode assumptions of evolutionary processes in terms of graph operations. The key evolutionary process characterizing biological networks is duplication and divergence (Wagner, 2001). Thus each evolutionary step is modeled by duplication of a network node (including its incident edges) and deletion of some of the incident edges. Such models have been elucidated and validated in several biological studies (Chung *et al.*, 2003; Vázquez *et al.*, 2003). In this work we use the Duplication-Mutation with Complementarity (DMC) model, that has been found to fit PPI networks better than other commonly used network growth models (Middendorf *et al.*, 2005; Navlakha and Kingsford, 2011).

Similar to reconstruction algorithms to infer evolutionary history of sequences, we can use a network growth model to obtain principled model-based reconstruction of ancestral networks. Assuming such a generative model, ancestral reconstruction

seeks to find the most likely sequence of networks that yields the extant network. This entails inferring the order in which nodes duplicate and edges are lost at each step during evolution. Several algorithms have been designed for ancestral network reconstruction. An algorithm for maximum likelihood ancestral reconstruction based on the DMC model, called ReverseDMC, was developed by Navlakha and Kingsford (2011). ReverseDMC greedily (by maximizing the likelihood of that single step) chooses an *anchor* node that is duplicated, at each step of evolution.

ReverseDMC uses only extant network topology to infer ancestral networks. Variants that can use additional biological information of the extant proteins, when available, for ancestral reconstruction have also been proposed. Such additional information include protein duplication history (Li *et al.*, 2013; Jasra *et al.*, 2015) and evolutionary periods of proteins (Zhang *et al.*, 2017). Other techniques for ancestral network reconstruction include the use of graphical models (Pinney *et al.*, 2007), and parsimony-based approaches that find one or more ancestral reconstructions with the minimum number of interaction gain/loss events (Patro *et al.*, 2012; Patro and Kingsford, 2013). These methods also use the gene duplication history and extant networks of multiple species during ancestral network reconstruction. Most of these methods, including ReverseDMC, yield only one evolutionary history, which is obtained by optimizing a mathematical criterion (like likelihood). In many applications it is useful to obtain multiple optimal and near-optimal histories to explore their biological relevance, through alternative criteria.

In this paper, we develop an Integer Linear Programming (ILP) solution for maximum likelihood reconstruction of ancestral PPI networks, using only extant network information. We use indicator variables to determine anchor and duplicated nodes at each step of evolution. Conditions imposed by the DMC model are formulated as linear constraints on each consecutive pair of networks during evolution. By construction, our algorithm can find the optimal solution, i.e., a solution that maximizes DMC-model based likelihood.

It is not known whether this problem is polynomial-time solvable. However, it appears to be unlikely, since the number of possible histories grows exponentially with each step. The advantage of an ILP framework is that it can leverage accurate and efficient heuristics, that are being steadily improved by the optimization community with readily available implementations in state-of-the-art general-purpose solvers (Gurobi, 2015). These improvements can automatically enhance the solution quality for the ancestral reconstruction problem. Another advantage of using ILP heuristics is that they can find multiple optimal and near-optimal solutions during their search of the solution space. Thus, they yield multiple reconstructions that can be examined for their biological relevance.

In experiments with synthetic datasets, our ILP solution obtains reconstructions with higher likelihood than those from ReverseDMC, which also shows that the greedy heuristic for this problem is not optimal. We evaluate our algorithm on two real biological networks, that contain protein-protein interactions from the families of bZIP transcription factors and Commander complex. Our ILP obtains solutions with higher likelihood on both these networks. We also examine the biological relevance of the results by comparing the inferred node arrival times as well as the chosen duplicated nodes at each evolutionary step, in reconstructions from ReverseDMC and ILP. By corroboration with independent biological evidence, we find that ILP produces better results.

2 Problem Statement

Given a network G_t at time t, and a model of evolution $\mathcal{M}$ that specifies a series of operations that generates G_{g+1} from G_g, we want to find the most probable sequence of networks $G_S = G_1, G_2, \ldots, G_{t-1}$:

$$G_S^* = \underset{G_S}{\operatorname{argmax}}(P(G_t|G_{t-1}) \ldots P(G_2|G_1) \,|\, \mathcal{M}, G_t) \qquad (1)$$

We now describe the model that we use and how likelihood is computed for the model, as given in Navlakha and Kingsford (2011).

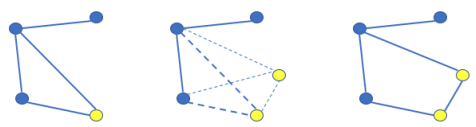

Fig. 1. DMC Model. Left: Yellow anchor node selected. Middle: Anchor node is duplicated, with edges to all neighbors. Right: Some edges to neighbors are deleted (with probability $q_{\text{mod}}/2$), edge between the duplicated nodes retained with probability q_{con}.

The duplication-mutation with complementarity (DMC) model assumes the first network to be a simple, connected two–node graph, has two parameters q_{con} and q_{mod}, and network evolution, from any network G_g to G_{g+1}, proceeds as follows (see fig. 1):

1. An anchor node u in G_g is selected at random and duplicated to form node v. Initially v is connected to all neighbors of u and to no other nodes.
2. For each neighbor x of u (x is also a neighbor of v), the connecting edge (u, x) or (v, x) is modified with probability q_{mod}; if the edge is to be modified, then with equal probability, either edge (u, x) or (v, x) is deleted.
3. Edge (u, v) is added with probability q_{con}.

Since each time-step adds a node we denote each network by the number of nodes contained in it: G_g is a network with g nodes.

Let e_{uv} denote the edge between the anchor (u) and duplicated node (v), that is set to 1 if the edge exists and is 0 otherwise. From step 2 of the DMC model, the probability that u and v share a particular neighbor is $(1 - q_{\text{mod}})$ and the probability that a node x is a neighbor of u and not of v or a neighbor of v and not of u is $q_{\text{mod}}/2$. Let $N(u)$ denote the neighbors of u, the intersection $N(u) \cap N(v)$ is the set of common neighbors of u and v and the symmetric difference $N(u)\Delta N(v)$ is the set of nodes that are neighbors of either u or v but not both. Then, given u and v are the anchor and duplicated nodes respectively in G_g, we have, ignoring constant terms:

$$\begin{aligned} \log P(G_g|G_{g-1}, \mathcal{M}) &= (e_{uv} \log q_{\text{con}} + (1 - e_{uv}) \log(1 - q_{\text{con}}) \\ &+ \sum_{N(u)\cap N(v)} (1 - q_{\text{mod}}) \\ &+ \sum_{N(u)\Delta N(v)} q_{\text{mod}} \end{aligned}$$

Once u and v are identified, G_{g-1} can be reconstructed by removing node v and adding edges between u and each node in $N(u) - (N(u) \cap N(v))$, since these edges were present before step 2 of the DMC model. Note that u and v are indistinguishable in G_g: either one of them may be deleted to form G_{g-1} and the addition of edges follows *mutatis mutandis*. In the following we will refer to the pair of nodes u, v in G_g as *duplicated nodes* and u in G_{g-1} as the *anchor node*.

3 ILP-based Solution

To recover the entire sequence G_S, given the extant network G_t, we have to identify the following:

- Anchor nodes in each of the networks $G_2, \ldots, G_{t-1}$,
- Duplicated nodes in each of the networks $G_3, \ldots, G_t$,
- Edges in each of the networks $G_3, \ldots, G_{t-1}$.

We will construct an Integer Linear Program (ILP) to obtain the solution. For each graph, $G_2, \ldots, G_t$, we will use binary edge indicators e_{ijg} that denote presence or absence of an edge and binary node indicators $x_{ig}, y_{ig}, z_{ig}, a_{ig}$. Subscripts i, j refer to nodes and g refers to network G_g that has nodes $1, \ldots, g$. We will set x_{ig} to 1 if the i^{th} node in G_g is a duplicated node and a_{ig} to 1 if the i^{th} node in G_g is an anchor node. To identify a common neighbor of the duplicated nodes, we will use the indicator y_{ig} and to identify a neighbor of either one of the duplicated nodes (but not both), we will use the indicator z_{ig}. Note that $e_{ijg},\ \forall i, j$ are known in networks G_2 and G_t and unknown in all the other networks. All the binary node indicators are unknown in all the networks.

The log of the probability in equation 1 can now be expressed as:

$$\begin{aligned} lP &= \sum_{g=1}^{t} \Big(\sum_{i=1}^{g} \sum_{j=1}^{g} (e_{ijg} x_{ig} x_{jg} \log q_{\text{con}} \\ &+ (1 - e_{ijg}) x_{ig} x_{jg} \log(1 - q_{\text{con}}) \Big) \\ &+ \sum_{k=1}^{g} y_{kg} (1 - q_{\text{mod}}) + \sum_{k=1}^{g} z_{kg} q_{\text{mod}} \end{aligned}$$

Thus we want to maximize lP subject to all the constraints (2 to 23 below) posed by the extant graph and the model, which we shall now describe.

3.1 Anchors, Duplicated Nodes and Neighbors

Each network, except G_2, has exactly 2 duplicated nodes:

$$\sum_{i=1}^{g} x_{ig} = 2, \quad \forall g \in \{3, \ldots, t\} \tag{2}$$

Each network, except G_t, has exactly 1 anchor node:

$$\sum_{i=1}^{g} a_{ig} = 1, \quad \forall g \in \{2, \ldots, t-1\} \tag{3}$$

The product $e_{ijg} x_{ig}$ is 1 if and only if the i^{th} node is a duplicated node and there is an edge from the j^{th} node to the i^{th} node. If the k^{th} node is a common neighbor there should be exactly 2 edges to the duplicated nodes in the network. Since there are only 2 duplicated nodes per network, for the k^{th} node, the sum $\sum_{i=1}^{g} e_{ikg} x_{ig}$ can take only three values: 0,1 or 2. For values 0 and 1, constraint 4 sets $y_{kg} = 0$ and for value 2, constraints 4 and 5 set $y_{kg} = 1$.

$$2y_{kg} \leq \sum_{i=1}^{g} e_{ikg} x_{ig}, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{4}$$

$$y_{kg} \geq \sum_{i=1}^{g} e_{ikg} x_{ig} - 1, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{5}$$

To identify a neighbor of one of the duplicated nodes, but not both, i.e. to set z_{kg}, there should be exactly 1 edge to the duplicated nodes in the network. We also have to ensure that one of the duplicated nodes, which may also satisfy this criterion if the duplicated nodes have an edge between them, is not selected. We can pose these constraints using an auxiliary binary node variable w_{kg}:

$$w_{kg} + 2y_{kg} = \sum_{i=1}^{g} e_{ikg} x_{ig}, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{6}$$

$$z_{kg} \geq w_{kg} - x_{kg}, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{7}$$

$$z_{kg} \leq w_{kg}, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{8}$$

$$z_{kg} \leq 1 - x_{kg}, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{9}$$

Since there are only 2 duplicated nodes per network, for the k^{th} node, the sum $\sum_{i=1}^{g} e_{ikg} x_{ig}$ can take only three values: 0,1 or 2.

- If the value is 2, then constraints 4 and 5 ensure that $y_{kg} = 1$ and constraint 6 sets $w_{kg} = 0$ yielding $z_{kg} = 0$ through constraint 8.
- If the value is 1, then $w_{kg} = 1$ since constraints 4 and 5 ensure that $y_{kg} = 0$. In this case if $x_{kg} = 1$ then constraint 9 ensures that $z_{kg} = 0$ and if $x_{kg} = 0$ then constraint 7 ensures that $z_{kg} = 1$.
- Finally, if the value is 0, then $w_{kg} = 0$ (constraints 4, 5, 6) and $z_{kg} = 0$ through constraint 8.

We use another binary node variable n_{kg} to indicate a neighbor of a duplicated node, which may be a common neighbor or neighbor of either of the duplicated nodes:

$$n_{kg} = y_{kg} + z_{kg}, \quad \forall k, \forall g \in \{3, \ldots, t\} \tag{10}$$

3.2 Phantom Edges

During reconstruction, we have to learn the correspondence between nodes in G_g and nodes in the previous network G_{g-1} to set the values of the unknown edges. In particular, we want to associate the duplicated nodes in network G_g with the anchor node in G_{g-1}. To learn this association, we use indicator variables $P_{jg}^{i(g-1)}$ for pairs of nodes (i_{g-1}, j_g) where the subscript indicates the network to which the node belongs. Since these are edges that do not exist in the network, but are artificial constructions for our inference, we call them *phantom edges*. We can view them as directed edges to a network from the previous network. See fig. 2 for an illustration.

On each node j_g in a network, except in G_2, there must be exactly one incoming phantom edge from any of the nodes (i_{g-1})

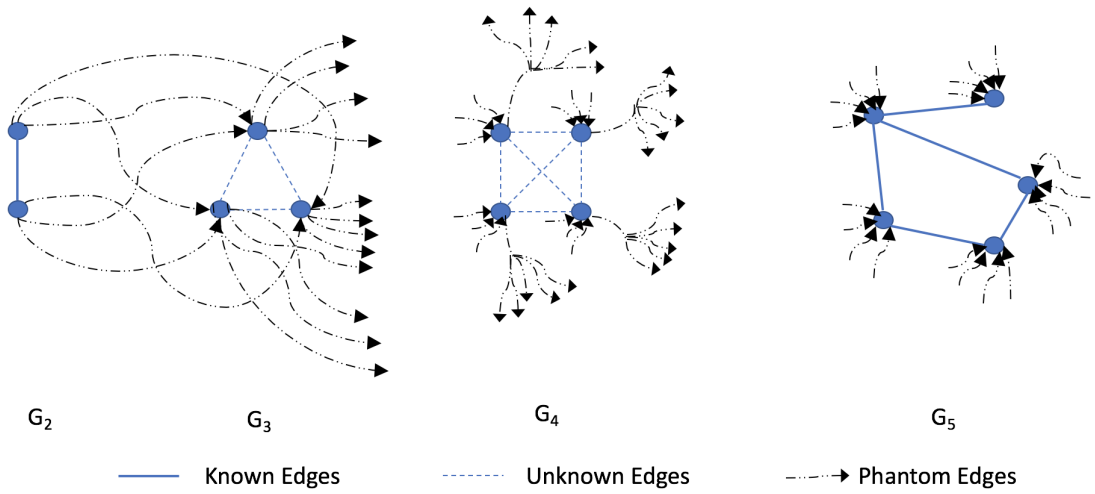

Fig. 2. Phantom edges between networks. Each node i_g of a network G_g is connected to all the nodes j_g in G_g through phantom edges $P_{jg}^{i(g-1)}$. Not all phantom edges shown.

in the previous network:

$$\sum_{i_{g-1}=1}^{g-1} P_{jg}^{i(g-1)} = 1, \quad \forall j_g \forall g \in \{3, \ldots, t\} \tag{11}$$

From each node (i_{g-1}) in the (previous) network, except from G_t, there must be at least 1 and at most 2 outgoing phantom edges. Anchor nodes will have 2 phantom edges and all other nodes will have only 1:

$$\sum_{j_g=1}^{g} P_{jg}^{i(g-1)} \geq 1, \quad \forall i_{g-1} \forall g \in \{3, \ldots, t\} \tag{12}$$

$$\sum_{j_g=1}^{g} P_{jg}^{i(g-1)} \leq 2, \quad \forall i_{g-1} \forall g \in \{3, \ldots, t\} \tag{13}$$

3.3 Edge Reconstruction

We now add the final set of constraints for edges in all the ancestral networks that are determined by the model and edges in the extant network. This is done by *mapping* edges from G_g to G_{g-1} for which we will use the phantom edges. The known edges in the extant network shall be mapped backwards up to the first graph G_2. We have to ensure the following three conditions:

1. An edge between duplicated nodes should not be mapped to any edge in the previous network since the duplicated nodes are from a single anchor node.
2. An edge (x_g, n_g) between a duplicated node x_g and its neighbor n_g in network G_g should be mapped to an edge (a_{g-1}, n_{g-1}) between the anchor a_{g-1} and its neighbor n_{g-1} in network G_{g-1}.
3. Any other edge should be mapped back to a unique edge in the previous network and there should be no other unmapped edge in the previous network.

To set these constraints, we will use three variables defined as follows. A binary indicator variable, for two nodes i_g and j_g in G_g, is defined as

$$S_{ijg}^{1} = \sum_{k=1}^{g} a_{k(g-1)} P_{kg}^{i(g-1)} P_{kg}^{j(g-1)}.$$

It is non-zero if and only if there are two phantom edges from an anchor node k_{g-1} in G_{g-1} to i_g and j_g in G_g. For each edge (i, j), each term in S_{ijg}^{1} is the product of $a_{k(g-1)}, P_{kg}^{i(g-1)}, P_{kg}^{j(g-1)}$. This term has value 1 iff $a_{k(g-1)} = P_{kg}^{i(g-1)} = P_{kg}^{j(g-1)} = 1$ which creates a mapping from nodes i, j to the anchor node in the previous network. See fig. 3.

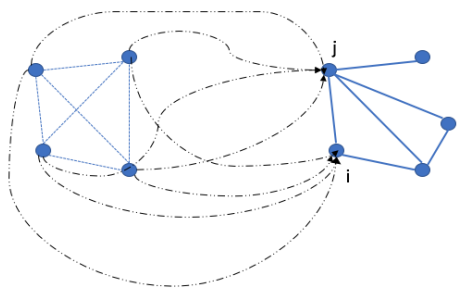

Fig. 3. For each pair of nodes (i, j), we use phantom edges to find the appropriate mapping. Variables $S_{ijg}^{1}, S_{ijg}^{2a}, S_{ijg}^{2b}, T_{ijg}$ encode different possible conditions all of which are not true at the same time. S_{ijg}^{1} is used to map i, j to a single anchor node, $S_{ijg}^{2a}, S_{ijg}^{2b}$ are used to map i, j to an anchor node and its neighbor and T_{ijg} is used for all other cases.

Another binary indicator variable, for two nodes i_g and j_g in G_g, is defined as

$$S_{ijg}^{2a} = \sum_{l,k=1}^{g-1} a_{l(g-1)}(1 - a_{k(g-1)}) P_{jg}^{k(g-1)} P_{ig}^{l(g-1)} e_{lk(g-1)}.$$

It is non-zero if an only if there are two phantom edges from an anchor node $a_{l(g-1)}$ and its neighbor $(1 - a_{k(g-1)})$ connecting

them respectively to i_g and j_g in G_g and there is an edge $e_{lk(g-1)}$ in G_{g-1}. For a symmetric condition, for phantom edges from an anchor node $a_{l(g-1)}$ and its neighbor $(1 - a_{k(g-1)})$ connecting them respectively to j_g and i_g in G_g, we define another binary indicator variable, for two nodes i_g and j_g in G_g, as

$$S^{2b}_{ijg} = \sum_{l,k=1}^{g-1} a_{l(g-1)}(1 - a_{k(g-1)}) P^{k(g-1)}_{ig} P^{l(g-1)}_{jg} e_{lk(g-1)}.$$

Each term in the sums S^{2a}_{ijg} and S^{2b}_{ijg} is used to create a mapping from nodes i, j to an anchor node and its neighbor in the previous network. See fig. 3.

Finally, another binary indicator variable, for two nodes i_g and j_g in G_g, is defined as

$$T_{ijg} = \sum_{l,k=1}^{g-1} P^{l(g-1)}_{ig} P^{k(g-1)}_{jg} e_{lk(g-1)}.$$

It is non-zero if and only if there are two phantom edges from (any) nodes k_{g-1} and l_{g-1} in G_{g-1} to i_g and j_g in G_g respectively and there is an edge $e_{lk(g-1)}$. Each term in T_{ijg} is a product of phantom nodes incoming at i and j in G_g and the edge $e_{lk(g-1)}$ in the previous network G_{g-1}, which when set to 1 creates a mapping from edge $(i, j) \in G_g$ to edge $(l, k) \in G_{g-1}$. See fig. 3.

We set the constraints for each pair of nodes (i_g, j_g) in graph G_g based on node indicators for duplicated nodes (x_{ig}) and neighbor nodes (n_{ig}):

- If both (i_g, j_g) are duplicated nodes, i.e. $x_{ig}x_{jg} = 1$, then we have to set $S^1_{ijg} = 1$ to ensure that duplicated nodes connect to an anchor node in the previous network. Other indicators, $S^{2a}_{ijg} = S^{2b}_{ijg} = T_{ijg} = 0$ to ensure that no edge in G_{g-1} is mapped to an edge, if any, between i_g and j_g. See fig. 4.

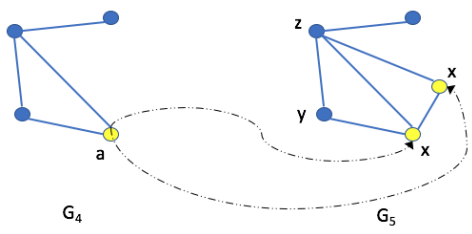

Fig. 4. If both (i_g, j_g) are duplicated nodes (denoted by x), then $S^1_{ijg} = 1$ shall connect the duplicated nodes to a single anchor node (denoted by a) in the previous network.

- If the nodes (i_g, j_g) are such that one of them is a duplicated node and the other a neighbor, i.e. $x_{ig}n_{ig} = 1$ or $x_{jg}n_{ig} = 1$, then we set $S^1_{ijg} = 0$ so the anchor node in the previous network does not connect to this pair through any phantom edges, and we set $S^{2b}_{ijg} = x_{jg}n_{ig}, S^{2a}_{ijg} = x_{ig}n_{jg}$ to ensure that phantom edges connect the anchor and its neighbor in the previous graph to nodes (i_g, j_g).
  Note that there may not be an edge between (i_g, j_g), if j_g is a neighbor to the other duplicated node and not i_g as shown in fig. 5. This should still set the above constraints since both the duplicated nodes map to the anchor. We set $T_{ijg} = 1$ to ensure that there is exactly one edge between (l_{g-1}, k_{g-1}) and

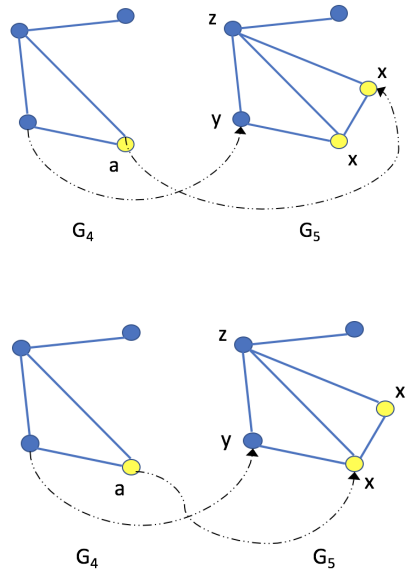

Fig. 5. A duplicated node (x) and a neighbor (y or z) must connect to an anchor (a) and its neighbor in the previous network. This is done through the variables $S^{2a}_{ijg}, S^{2b}_{ijg}$. Above: A duplicated node and neighbor of the other duplicated node, Below: A duplicated node and its own neighbor. Both have to be mapped to the same two nodes in the previous network.

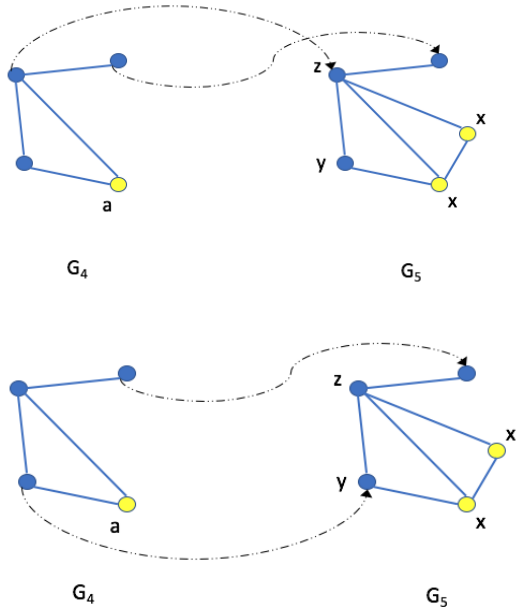

Fig. 6. Above: An edge between non-duplicated nodes is mapped back to an edge in the previous network. Below: If there is no edge between a pair of non-duplicated nodes, there should be no edge in the mapped nodes in the previous network.

$T_{ijg} \geq e_{ijg}$ since there may or may not be an edge between (i_g, j_g).
Note that this and the previous cases are mutually exclusive since n_{ig} and x_{ig} are never both set to 1 for the same node.

- If both the above cases are not true, i.e. $x_{ig}x_{jg} = x_{ig}n_{ig} = x_{jg}n_{ig} = 0$, then we set $S^1_{ijg} = S^{2b}_{ijg} = S^{2a}_{ijg} = 0$ since we do not want an edge between (i_g, j_g) to map to any edge connecting to an anchor in the previous network and we set $T_{ijg} = e_{ijg}$ to ensure that there is a single edge (l_{g-1}, k_{g-1}) if $e_{ijg} = 1$. If $e_{ijg} = 0$, then this ensures there is no edge in the previous network mapped to (i_g, j_g). See fig. 6.

The above three sets of conditions are incorporated in the following constraints:

$$S^1_{ijg} = x_{ig}x_{jg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{14}$$

$$S^{2a}_{ijg} = x_{ig}n_{jg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{15}$$

$$S^{2b}_{ijg} = x_{jg}n_{ig}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{16}$$

We define an auxiliary binary variable P_{ijg} that is set to 0 if $S^{2a}_{ijg} = 0$ and $S^{2b}_{ijg} = 0$ and 1 otherwise (i.e. the logical OR); also, we set $Q_{ijg} = x_{ig}x_{jg}$:

$$P_{ijg} \geq S^{2a}_{ijg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{17}$$

$$P_{ijg} \geq S^{2b}_{ijg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{18}$$

$$P_{ijg} \leq S^{2a}_{ijg} + S^{2b}_{ijg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{19}$$

Variables T_{ijg} and e_{ijg} are set using P_{ijg} and Q_{ijg}:

$$T_{ijg} \geq P_{ijg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{20}$$

$$T_{ijg} \leq 1 + P_{ijg} - Q_{ijg}, \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{21}$$

$$e_{ijg}(1 - Q_{ijg}) \leq T_{ijg}(1 - Q_{ijg}), \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{22}$$

$$e_{ijg}(1 - P_{ijg}) \geq T_{ijg}(1 - P_{ijg}), \quad \forall i_g, j_g, \forall g \in \{3, \ldots, t\} \tag{23}$$

- If $Q_{ijg} = x_{ig}x_{jg} = 1$, then constraints 21 and 22 ensure that $T_{ijg} = P_{ijg} = 0$ since both S^{2a}_{ijg} and S^{2b}_{ijg} are 0. If $P_{ijg} = 0, Q_{ijg} = 1$, then constraint 22 is void and constraint 23 ensures that $e_{ijg} \geq T_{ijg}$.
- If $Q_{ijg} = x_{ig}x_{jg} = 0$ and $P_{ijg} = 1$ (i.e. either S^{2a}_{ijg} or S^{2b}_{ijg} is 1 which is only possible if $x_{ig}n_{ig} = 1$ or $x_{jg}n_{ig} = 1$) then constraint 20 ensures that $T_{ijg} = 1$. If $P_{ijg} = 1, Q_{ijg} = 0$, then constraint 23 is void and constraint 22 ensures that $e_{ijg} \leq T_{ijg}$.
- If $Q_{ijg} = x_{ig}x_{jg} = 0$ and $P_{ijg} = 0$, then constraints 20 and 21 do not impose any value on T_{ijg}. If $P_{ijg} = 0, Q_{ijg} = 0$, then constraints 22 and 23 ensure that $e_{ijg} = T_{ijg}$.

Finally, we set $e_{ijg} = e_{jig} \forall i_g, j_g, \forall g \in \{2, \ldots, t\}$ to ensure that the edges are undirected.

3.4 Linearization

The constraints as described above have terms that are products of binary variables and sums of such products. A constraint $y = x_1x_2, \ldots, x_n$ where each variable is binary is equivalent to the following $n + 1$ constraints: $y \geq x_1, y \geq x_2, \ldots y \geq x_n, y \leq x_1 + x_2 + \ldots + x_n - (n-1)$. Sums of products can be decomposed using auxiliary binary variables. For example, $y = x_1x_2 + x_3x_4$ can be expressed as $y = z_1 + z_2, z_1 = x_1x_2, z_2 = x_3x_4$ and further linearized using the previous rule.

3.5 Multiple Solutions

Since ILP is, in general, NP-hard, optimal solutions for very large networks may not be found in polynomial time. However, many heuristics have been developed to find multiple near-optimal solutions, e.g., see Wallace (2010), with efficient software implementations (Gurobi, 2015). These heuristics enable us to find multiple solutions and examine their biological relevance.

4 Experiments

4.1 Simulations

We simulated 1215 extant networks with number of nodes in the extant network varying from 6 to 10. For each network, evolution is simulated following the DMC model starting from an initial network of two connected nodes. The DMC model requires two parameters q_{con} and q_{mod}. For each simulation, each parameter is randomly chosen from the closed interval $[0.1, 0.9]$, rounded to one decimal. We reconstruct the network sequence using ReverseDMC, the Greedy approach of (Navlakha and Kingsford, 2011) and our ILP.

Likelihood Comparison

Table 1 shows that out of 1215 simulations, there were no simulations where solutions from ILP had a lower likelihood than that of ReverseDMC. Since these are relatively small networks, both ReverseDMC and ILP were able to find optimal solutions in many cases. The fact that ILP could find 274 solutions with higher likelihood shows that ReverseDMC is not guaranteed to find optimal solutions. Table 2 shows the summary statistics of the increase in log-likelihood due to ILP, compared to ReverseDMC solutions. The increase can range from 0.04 to 2.33, even in these relatively small networks.

Total	Equal	ILP	ReverseDMC
1215	941	274	0

Table 1. Number of simulations where the log-likelihood of the reconstructed solutions were equal for both methods (2nd column), higher for ILP (3rd column) or higher for ReverseDMC (4th column).

Total	Mean (SD)	Median	Maximum	Minimum
274	0.68 (0.43)	0.54	2.33	0.04

Table 2. Summary statistics of increase in log-likelihood among the 274 simulations where ILP reconstructions had higher likelihood than ReverseDMC reconstructions. SD: standard deviation.

4.2 Real Networks

We reconstruct the history of two protein-protein interaction networks using both ReverseDMC and ILP algorithms. In both algorithms, we assume $q_{con} = 0.7, q_{mod} = 0.4$.

We evaluate the biological relevance of the results in two ways. First, we compare the node arrival times of the reconstructions following the procedure described in Navlakha and Kingsford (2011). The key idea is to estimate the protein arrival time using available ortholog information, with the assumption that proteins that arrive earlier in history have higher number of orthologs.

Thus, the list of proteins in the extant network in descending order of number of orthologs is considered to be the 'true' node arrival order (A_T). We determine the number of orthologs for each protein using OrthoDB (Kriventseva *et al.*, 2018), by counting the number of genes at the highest level at which ortholog information was available for all the proteins in the networks (vertebrata for bZIP and metazoa for Commander). The reconstruction history of both Greedy and ILP identifies the removed node at each step: this provides the reconstructed node arrival order (A_R) for each algorithm. A_T and A_R are compared using Kendall's Tau (Kendall, 1945) that measures correlation between two ranked lists (definition given in appendix). Higher values indicate better correlation.

Our second evaluation is based on the sequence similarities between all the inferred anchors and duplicated nodes. Since at each time step in evolution (by the DMC model) the anchor gene (a) duplicates into another gene (d), we expect the pairwise similarity between a and d to be higher than the pairwise similarity between a and the remaining genes at that time step. Given the the extant network G_t and its reconstructed evolutionary history: $\hat{G}_S = \hat{G}_3, \ldots, \hat{G}_{t-1}, G_t$, along with chosen anchors and duplicated nodes in each network, we compute a score $\rho(\hat{G}_i)$ for each network in $\hat{G}_i \in \hat{G}_S$, using pairwise sequence similarity (Needleman and Wunsch, 1970) between the chosen anchor node protein and the duplicated node protein. The final score for the reconstruction, that we call Anchor-Duplicate Similarity Score (ADSS), is given by $\sum_{\hat{G}_i \in \hat{G}_S} \rho(\hat{G}_i)/(t-2)$, where we normalize by the number of networks in $\hat{G}_S$. $\hat{G}_2$ is not considered since in the first evolutionary step (from $\hat{G}_1$ to $\hat{G}_2$) there is only one gene that duplicates and there are no other genes to compare with. Thus given two reconstructions of the same extant network, higher ADSS indicates better choice of anchor and duplicate nodes in the reconstruction.

bZIP Transcription Factors

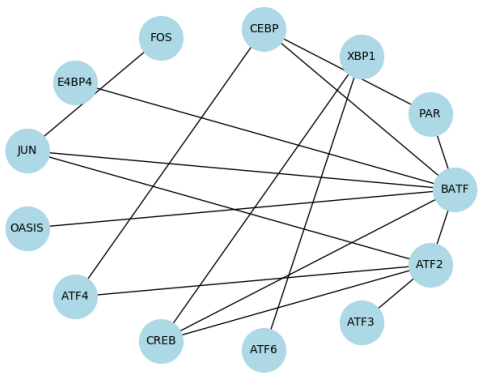

Fig. 7. Extant bZIP network used in our experiment.

The basic-region leucine zipper (bZIP) transcription factors are a protein family involved in many cellular processes including the regulation of development, metabolism, circadian rhythm, and response to stress and radiation (Amoutzias *et al.*, 2006; Pinney *et al.*, 2007). The interactions between these proteins are strongly mediated by their coiled-coil leucine zipper domains and so, the strength of these interactions can be accurately predicted using just sequence information (Fong *et al.*, 2004). With the method of Fong *et al.* (2004), Pinney *et al.* (2007) constructed extant networks on a set of bZIP proteins for multiple species. We took the *H. sapiens* network and merged subunits for the same protein into one node, to obtain the extant network used in our experiment (fig. 7).

Algorithm	Log-Likelihood	Kendall's Tau	ADSS
ReverseDMC	-21	-0.23	-1901.55
ILP	-19.6	0.26	-1797.11

Table 3. Likelihood, Node Arrival Time Accuracy and ADSS for Ancestral Reconstruction of the bZIP Network

Table 3 shows the Likelihood, Node Arrival Time Accuracy (measured by Kendall's Tau) and ADSS for Ancestral Reconstruction of the bZIP Network by both ReverseDMC and ILP. With respect to all three metrics, the solution obtained by ILP is better than that of ReverseDMC. Table 4 shows the order of arrival of proteins inferred by the reconstructions from ReverseDMC and ILP. Sequence-based phylogenetic analysis of bZIP transcription factors by Amoutzias *et al.* (2006) revealed a highly conserved ancient core network containing proteins JUN, FOS and ATF3, that provides additional evidence of the correctness of our reconstruction. In table 4 we observe that these three proteins appear early in the order inferred by ILP (before the seventh step) while JUN and ATF3 arrive after the seventh step in the order inferred by ReverseDMC.

Time step	ReverseDMC	ILP
2	FOS, CREB	ATF2, ATF6
3	ATF6	FOS
4	BATF	ATF3
5	ATF4	CREB
6	ATF2	JUN
7	E4BP4	CEBP
8	JUN	BATF
9	ATF3	PAR
10	CEBP	ATF4
11	PAR	E4BP4
12	XBP1	OASIS
13	OASIS	XBP1

Table 4. Arrival order of anchor proteins in the bZIP network, at each step of evolution, based on reconstructions from ReverseDMC and ILP.

Commander Network

Commander is a multiprotein complex that is broadly conserved across vertebrates and is involved in several roles including pro-inflammatory signaling and vertebrate embryogenesis (Mallam and Marcotte, 2017). A well characterized sub-complex of Commander, CCC, made of COMMD1, CCDC22, CCDC93 and C16orf62, is known to be involved in endosomal protein trafficking (Bartuzi *et al.*, 2016; Mallam and Marcotte, 2017). Defects in the Commander complex are associated with developmental disorders (Mallam and Marcotte, 2017; Liebeskind *et al.*, 2018). Reconstructing the evolutionary history of interactions in the complex can shed light on the conservation and stability of the proteins and their interactions, which in turn can aid understanding of the sources of dysfunction of the complex. We use the network discussed in (Liebeskind *et al.*, 2018), shown in fig. 8, as the extant network for ancestral reconstruction.

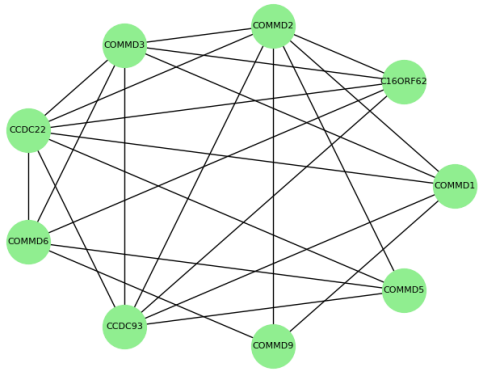

Fig. 8. Extant Commander network used in our experiment.

Algorithm	Log-Likelihood	Kendall's Tau	ADSS
ReverseDMC	-17.49	-0.27	-2873.25
ILP	-16.17	0.01	-1649.64

Table 5. Likelihood, Node Arrival Time Accuracy and ADSS for Ancestral Reconstruction of the Commander Network

Time step	ReverseDMC	ILP
2	COMMD6, COMMD9	COMMD3, COMMD9
3	C16ORF62	COMMD1
4	COMMD2	COMMD6
5	CCDC93	COMMD2
6	CCDC22	COMMD5
7	COMMD5	CCDC93
8	COMMD1	CCDC22
9	COMMD3	C16ORF62

Table 6. Arrival order of anchor proteins in the commander network, at each step of evolution, based on reconstructions from ReverseDMC and ILP.

Table 5 shows the Likelihood, Node Arrival Time Accuracy (measured by Kendall's Tau), and ADSS for ancestral reconstruction by both ReverseDMC and ILP. On this network too, on all three metrics, the solution obtained by ILP is better than that of ReverseDMC. Table 6 shows the order of arrival of proteins inferred by the reconstructions from ReverseDMC and ILP. Among all the commander proteins, COMMD1 is the best studied and is found to be highly conserved with multiple key functions (Riera-Romo, 2018). Indeed, in OrthoDB, COMMD1 has the maximum number of orthologs, among these proteins. In the reconstruction by ILP, COMMD1 is seen to arrive early, at the third step, while in the reconstruction from ReverseDMC it arrives only at the eighth step of evolution.

5 Conclusion

We presented an Integer Linear Programming (ILP) based solution for maximum-likelihood reconstruction of the evolution of a PPI network using the Duplication-Mutation with Complementarity (DMC) model. We use indicator variables to determine anchor and duplicated nodes and derived the conditions imposed by the DMC model as linear constraints on each network, and on each consecutive pair of networks, at each step during evolution. By construction, the ILP can find the optimal solution and heuristics from general-purpose ILP solvers can be used to find multiple optimal and near-optimal solutions efficiently.

We compared the solutions obtained by our ILP with those from ReverseDMC (Navlakha and Kingsford, 2011), the previous best algorithm for this problem. On simulated data, we found that ILP always obtains solutions that are of equal or higher likelihood than those from ReverseDMC. We evaluated both the algorithms on two real PPI networks, containing proteins from the bZIP transcription factors and Commander complex respectively. On both the networks, solutions from our ILP had higher likelihood and were in better agreement with independent biological evidence from ortholog information and sequence similarity.

This is the first ILP solution to a model-based network reconstruction problem and the presented framework may be useful for other network models as well. The ILP framework could be generalized to handle multiple input networks as well as to take into account additional information, such as gene duplication histories. A limitation of our solution is that it can take considerably long to find reconstructions for very large networks. However, the ILP framework can yield deeper insights into the structure of the problem and specialized heuristics could be developed to obtain more efficient and scalable solutions.

Acknowledgements

Part of the work was performed during XZ's visit at the Simons Institute for the Theory of Computing at University of California, Berkeley. We thank Rob Patro for sharing the bZIP network data used in their publication (Patro and Kingsford, 2013) which is originally from (Pinney *et al.*, 2007).

Funding

V.R. was supported by Singapore Ministry of Education Academic Research Fund [R-253-000-139-114]. C.K. was partially supported in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554 to C.K., by the US National Science Foundation (CCF-1256087, CCF-1319998) and by the US National Institutes of Health (R01GM122935). X.Z. was supported by grant #220558 from the Ragon Institute of MGH, MIT and Harvard.

Disclosure Statement

C.K. is a co-founder of Ocean Genomics.

References

Amoutzias, G., Veron, A., Weiner, J, I., Robinson-Rechavi, M., Bornberg-Bauer, E., Oliver, S., and Robertson, D. (2006). One Billion Years of bZIP Transcription Factor Evolution: Conservation and Change in Dimerization and DNA-Binding Site Specificity. *Molecular Biology and Evolution*, **24**(3), 827–835.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286**(5439), 509–512.

Bartuzi, P., Billadeau, D. D., Favier, R., Rong, S., Dekker, D., Fedoseienko, A., Fieten, H., Wijers, M., Levels, J. H., Huijkman, N., *et al.* (2016). CCC- and WASH-mediated endosomal sorting of LDLR is required for normal clearance of circulating LDL. *Nature Communications*, **7**, 10961.

Borenstein, E. and Feldman, M. W. (2009). Topological signatures of species interactions in metabolic networks. *Journal of Computational Biology*, **16**(2), 191–200.

Chung, F., Lu, L., Dewey, T. G., and Galas, D. J. (2003). Duplication models for biological networks. *Journal of Computational Biology*, **10**(5), 677–687.

Dutkowski, J. and Tiuryn, J. (2007). Identification of functional modules from conserved ancestral protein–protein interactions. *Bioinformatics*, **23**(13), i149–i158.

Fong, J. H., Keating, A. E., and Singh, M. (2004). Predicting specificity in bzip coiled-coil protein interactions. *Genome Biology*, **5**(2), R11.

Gurobi (2015). Gurobi optimizer reference manual. http://www.gurobi.com.

Jasra, A., Persing, A., Beskos, A., Heine, K., and De Iorio, M. (2015). Bayesian inference for duplication–mutation with complementarity network models. *Journal of Computational Biology*, **22**(11), 1025–1033.

Kendall, M. G. (1945). The treatment of ties in ranking problems. *Biometrika*, **33**(3), 239–251.

Kreimer, A., Borenstein, E., Gophna, U., and Ruppin, E. (2008). The evolution of modularity in bacterial metabolic networks. *Proceedings of the National Academy of Sciences*, **105**(19), 6976–6981.

Kriventseva, E. V., Kuznetsov, D., Tegenfeldt, F., Manni, M., Dias, R., Simão, F. A., and Zdobnov, E. M. (2018). OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Research*, **47**(D1), D807–D811.

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM.

Li, S., Choi, K. P., Wu, T., and Zhang, L. (2013). Maximum likelihood inference of the evolutionary history of a PPI network from the duplication history of its proteins. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, **10**(6), 1412–1421.

Liebeskind, B., Aldrich, R. W., and Marcotte, E. M. (2018). Ancestral reconstruction of protein interaction networks. *bioRxiv*, page 408773.

Mallam, A. L. and Marcotte, E. M. (2017). Systems-wide studies uncover commander, a multiprotein complex essential to human development. *Cell Systems*, **4**(5), 483–494.

Middendorf, M., Ziv, E., and Wiggins, C. H. (2005). Inferring network mechanisms: the *Drosophila melanogaster* protein interaction network. *Proceedings of the National Academy of Sciences*, **102**(9), 3192–3197.

Navlakha, S. and Kingsford, C. (2011). Network archaeology: uncovering ancient networks from present-day interactions. *PLoS Computational Biology*, **7**(4), e1001119.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**(3), 443–453.

Nichols, S. A., Dirks, W., Pearse, J. S., and King, N. (2006). Early evolution of animal cell signaling and adhesion genes. *Proceedings of the National Academy of Sciences*, **103**(33), 12451–12456.

Patro, R. and Kingsford, C. (2013). Predicting protein interactions via parsimonious network history inference. *Bioinformatics*, **29**(13), i237–i246.

Patro, R., Sefer, E., Malin, J., Marçais, G., Navlakha, S., and Kingsford, C. (2012). Parsimonious reconstruction of network evolution. *Algorithms for Molecular Biology*, **7**(1), 25.

Pereira-Leal, J. B., Levy, E. D., and Teichmann, S. A. (2006). The origins and evolution of functional modules: lessons from protein complexes. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, **361**(1467), 507–517.

Pinney, J. W., Amoutzias, G. D., Rattray, M., and Robertson, D. L. (2007). Reconstruction of ancestral protein interaction networks for the bZIP transcription factors. *Proceedings of the National Academy of Sciences*, **104**(51), 20449–20453.

Riera-Romo, M. (2018). COMMD1: A multifunctional regulatory protein. *Journal of Cellular Biochemistry*, **119**(1), 34–51.

Vázquez, A., Flammini, A., Maritan, A., and Vespignani, A. (2003). Modeling of protein interaction networks. *Complexus*, **1**(1), 38–44.

Vidal, M., Cusick, M. E., and Barabási, A.-L. (2011). Interactome networks and human disease. *Cell*, **144**(6), 986–998.

Wagner, A. (2001). The yeast protein interaction network evolves rapidly and contains few redundant duplicate genes. *Molecular Biology and Evolution*, **18**(7), 1283–1292.

Wallace, C. (2010). *Mixed integer programming heuristics*. Carnegie Mellon University.

Yamada, T. and Bork, P. (2009). Evolution of biomolecular networks - lessons from metabolic and protein interactions. *Nature Reviews Molecular Cell Biology*, **10**(11), 791.

Zhang, J., Yang, H., Song, H., and Zhang, Y. (2017). An improved archaeology algorithm based on integrated multi-source biological information for yeast protein interaction network. *IEEE Access*, **5**, 15893–15900.

Zhang, X. and Moret, B. M. (2010). Refining transcriptional regulatory networks using network evolutionary models and gene histories. *Algorithms for Molecular Biology*, **5**(1), 1.

Appendix

Kendall's Tau

We use the version that accounts for ties, given by $\tau = \frac{P-Q}{\sqrt{(P+Q+T)*(P+Q+U)}}$, where P is the number of concordant pairs, Q the number of discordant pairs, T the number of ties only in A_T, and U the number of ties only in A_R. If a tie occurs for the same pair in both A_T and A_R, it is not added to either T or U. Here we consider pairs of observatios $(x_i, y_i), (x_j, y_j)$ where $x_i, x_j \in A_T, y_i, y_j \in A_R$ and $i < j$. A pair $(x_i, y_i), (x_j, y_j)$ is concordant if the ranks of both elements agree, i.e., both $x_i < x_j$ and $y_i < y_j$; or both $x_i > x_j$ and $y_i > y_j$. A pair $(x_i, y_i), (x_j, y_j)$ is discordant if $x_i > x_j$ and $y_i < y_j$ or if $x_i < x_j$ and $y_i > y_j$. If $x_i = x_j$ or $y_i = y_j$, it is considered a tie.