Bioinformatics, 37(8), 2021, 1083–1092
doi: 10.1093/bioinformatics/btaa931
Advance Access Publication Date: 10 December 2020
Original Paper

Phylogenetics

Maximum likelihood reconstruction of ancestral networks by integer linear programming

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

[1]Department of Information Systems and Analytics, School of Computing, National University of Singapore, Singapore 117417, Singapore, [2]School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology, Atlanta 30308, GA, USA and [3]Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

*To whom correspondence should be addressed.

Associate Editor: Yann Ponty

Received on December 28, 2019; revised on September 17, 2020; editorial decision on October 17, 2020; accepted on October 23, 2020

Abstract

Motivation: The study of the evolutionary history of biological networks enables deep functional 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 (PPIs) 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. We prove the correctness of our solution that 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 data show that our ILP obtains solutions with higher likelihood than those from previous methods, and is robust to noise and model mismatch. We evaluate our algorithm on two real PPI networks, with proteins from the families of bZIP transcription factors and the Commander complex. On both the networks, solutions from our ILP have higher likelihood and are in better agreement with independent biological evidence from other studies.

Availability and implementation: A Python implementation is available at https://bitbucket.org/cdal/network-reconstruction.

Contact: vaibhav.rajan@nus.edu.sg

Supplementary information: Supplementary data are available at *Bioinformatics* online.

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 (PPIs), 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 PPIs (Dutkowski and Tiuryn, 2007), evolutionary trends of biosynthetic capacity loss in parasites (Borenstein and Feldman, 2009), regulatory network inference (Zhang and Moret, 2012) 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, which 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 (Jasra *et al.*, 2015; Li *et al.*, 2013), evolutionary periods of proteins (Zhang et al., 2017) and the node contents in ancestral networks (Jin *et al.*, 2013). 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. Some methods use the Preferential Attachment Model to reconstruct network growth history (Sreedharan *et al.*, 2019; Young *et al.*, 2019). 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 article, 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. We prove the correctness of our ILP formulation.

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, which are being steadily improved by the optimization community with readily available implementations in state-of-the-art general-purpose solvers. 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 near-optimal solutions during their search of the solution space. Thus, they yield multiple reconstructions that can be examined for their biological relevance. The ILP provides a theoretical framework for further algorithmic development of the maximum likelihood network reconstruction problem. We provide a Python implementation of our model using a general-purpose ILP solver (https://bitbucket.org/cdal/network-reconstruction).

In experiments with synthetic datasets, our ILP-based solution obtains reconstructions with higher likelihood than those from ReverseDMC, which also shows that the greedy heuristic for this problem is not optimal. Simulation studies show that our reconstruction algorithm is robust to noise in the extant network and mismatch in input model parameters. In addition to likelihood, we also compare the solutions reconstructed by ILP and ReverseDMC in terms of the node arrival order (measured by Kendall's Tau), duplication history (measured by distance between the reconstructed duplication tree and the true tree) and similarity of reconstructed ancestral networks to the true ones (measured by Kernel similarity of graphs). ILP solutions are found to be superior to ReverseDMC solutions on all these metrics in our simulations.

We evaluate our algorithm on two real biological networks that contain PPIs from the families of bZIP transcription factors and the Commander complex. Our ILP obtains solutions with higher likelihood, compared to those from ReverseDMC, 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. On both the networks, solutions from our ILP are in better agreement with independent biological evidence from ortholog information and sequence similarity.

2 Problem statement

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

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

where the probabilities are conditional on the model ℓ and the extant network G_t.

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

The DMC model assumes that G_2 is 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:

$$\log P(G_g|G_{g-1}, \ell) = e_{uv} \log q_{\text{con}} + (1 - e_{uv}) \log (1 - q_{\text{con}}) + \sum_{N(u)\cap N(v)} \log (1 - q_{\text{mod}}) + \sum_{N(u)\Delta N(v)} \log q_{\text{mod}}.$$

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

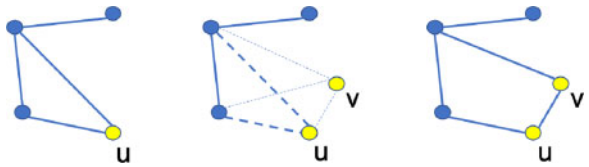

Fig. 1. DMC Model. Left: Yellow anchor node selected. Middle: Anchor node is duplicated, with edges to all neighbors. Right: Some edges to neighbors are deleted (with probability $q_{\text{mod}}/2$), edge between the duplicated nodes retained with probability q_{con}

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

3.1 Characterizing a reconstructed sequence

We first state a theorem that characterizes a sequence of networks that evolves following the DMC model. This forms the basis of our ILP and is also used in proving its correctness.

Let $G_g = \{N_g, E_g\}$ and $G_h = \{N_h, E_h\}$ be two networks, with N, E representing their sets of nodes and edges, respectively. Nodes and edges in $\{N_g, E_g\}$ are identified with the subscript g and are in lowercase (e.g. node $u_g \in N_g$, edge $e_h \in E_h$). We use e_ϕ as an indicator for a 'dummy edge' that does not exist in a network. An edge is represented by the pair of nodes it is incident on. For sets A, B, A/B denotes the set $A - B$.

Definition 3.1A pair of networks (G_g, G_h) is *DMC-evolvable* if G_h can be obtained from G_g through the DMC model of evolution in one step.

The theorem below (proof in Supplementary Appendix S1) characterizes DMC-evolvable pairs of networks from a combinatorial perspective without taking into account model parameters $q_{\text{con}}, q_{\text{mod}}$ (that determine the likelihood of the reconstruction).

Theorem 3.1A network G_g is *DMC-evolvable* into network G_h in one step iff:

1. $|N_h| = |N_g| + 1$
2. $\exists$ a function $f_N : N_h \to N_g$ and nodes $a_g \in N_g$ and $u_h, v_h \in N_h$ such that
   a. $f_N(u_h) = f_N(v_h) = a_g$
   b. The restriction of $f_N : N_h/\{u_h, v_h\} \to N_g/\{a_g\}$ is bijective.
3. The function $f_E : E_h \to E_g \cup \{e_\phi\}$ given by
   i. $f_E(x_h, y_h) = (f_N(x_h), f_N(y_h))$ and
   ii. $f_E(u_h, v_h) = e_\phi$, if $(u_h, v_h) \in E_h$

is well-defined and such that

a. $\forall$ node b_h that is a neighbor of either u_h or v_h (or both), and $b_h \neq u_h, b_h \neq v_h$, $f_E(u_h, b_h) = (a_g, f_N(b_h))$ and $f_E(v_h, b_h) = (a_g, f_N(b_h))$.

b. The restriction of $f_E : E_h/E_h^{uv} \to E_g$ is bijective, where $E_h^{uv} = \{(x_h, y_h) \mid y_h \in \{u_h, v_h\}, x_h \in N_h\}$, the set of edges incident on u_h and v_h.

Definition 3.2A sequence of networks $G_S = G_2, \ldots, G_t$ is *DMC-evolvable* if for every pair of consecutive networks (G_i, G_{i+1}), $i = \{2, 3, \ldots, t-1\}$, G_i is *DMC-evolvable* into G_{i+1}.

3.2 Our ILP

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. We denote the ith node in the gth graph by i_g and will omit the subscript in variable names, to avoid clutter, when the graph is clear from the context. 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 ith node in G_g is a duplicated node and a_{ig} to 1 if the ith 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:

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

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

3.3 Anchors, duplicated nodes and neighbors

Each network, except G_2, has exactly two 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 one 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 ith node is a duplicated node and there is an edge from the jth node to the ith node. If the kth 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 kth 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 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 two duplicated nodes per network, for the kth 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:

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

3.4 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 $P_{j_g}^{i_{g-1}}$ 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 Figure 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}) in the previous network:

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

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:

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

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

3.5 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

$$S_{ijg}^{1} = \sum_{k=1}^{g} a_{k(g-1)} P_{i_g}^{k_{g-1}} P_{j_g}^{k_{g-1}}.$$

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 $a_{k(g-1)}, P_{k_g}^{i_{g-1}}, P_{k_g}^{j_{g-1}}$. This term has value 1 iff $a_{k(g-1)} = P_{k_g}^{i_{g-1}} = P_{k_g}^{j_{g-1}} = 1$ which creates a mapping from nodes i, j to the anchor node in the previous network (see Fig. 3).

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

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

It is non-zero if and 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

$$S_{ijg}^{2b} = \sum_{l,k=1}^{g-1} a_{l(g-1)}(1 - a_{k(g-1)}) P_{i_g}^{k_{g-1}} P_{j_g}^{l_{g-1}} e_{lk(g-1)}.$$

Each term in the sums S_{ijg}^{2a} and S_{ijg}^{2b} 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

$$T_{ijg} = \sum_{l,k=1}^{g-1} P_{i_g}^{l_{g-1}} P_{j_g}^{k_{g-1}} e_{lk(g-1)}.$$

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).

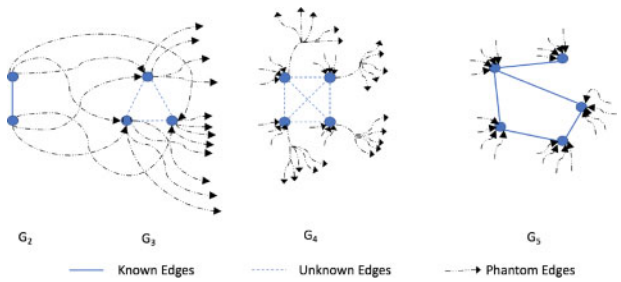

Fig. 2. Phantom edges between networks. Each node i_g of a network G_g is connected to all the nodes j_g in G_g through phantom edges $P_{ig}^{(g-1)}$. Not all phantom edges shown

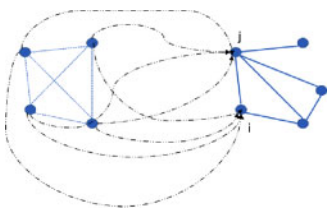

Fig. 3. For each pair of nodes (i, j), we use phantom edges to find the appropriate mapping. Variables $S_{ijg}^{1}, S_{ijg}^{2a}, S_{ijg}^{2b}, T_{ijg}$ encode different possible conditions all of which are not true at the same time

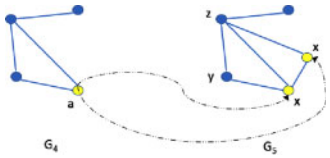

Fig. 4. If both (i_g, j_g) are duplicated nodes (denoted by x), then $S^1_{ijg} = 1$ shall connect the duplicated nodes to a single anchor node (denoted by a) in the previous network

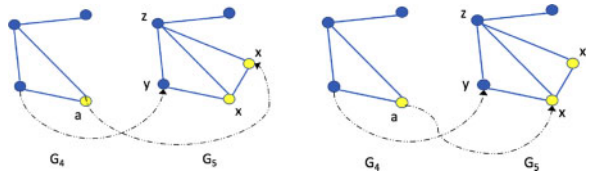

Fig. 5. A duplicated node (x) and a neighbor (y or z) must connect to an anchor (a) and its neighbor in the previous network. This is done through the variables $S^{2a}_{iig}, S^{2b}_{iig}$

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, $S^{2a}_{iig} = S^{2b}_{iig} = T_{iig} = 0$ 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).
- 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 $S^{2b}_{iig} = x_{ig}n_{ig}, S^{2a}_{iig} = x_{ig}n_{ig}$ 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 Figure 5. Since both the duplicated nodes map to the anchor, this constraint is set as required. We set $T_{ijg} = 1$ to ensure that there is exactly one edge between (l_{g-1}, k_{g-1}) and $T_{ijg} \geq e_{ijg}$ 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 $S^1_{ijg} = S^{2b}_{iig} = S^{2a}_{iig} = 0$ 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:

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

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

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

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

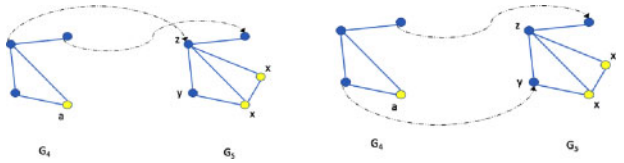

Fig. 6. Left: An edge between non-duplicated nodes is mapped back to an edge in the previous network. Right: If there is no edge between a pair of non-duplicated nodes, there should be no edge in the mapped nodes in the previous network

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

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

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

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

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

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

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

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

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

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

3.6 Proof of correctness

We prove (in Supplementary Appendix S2) the correctness of the ILP:

Theorem 3.2A feasible solution to the above ILP yields a DMC-evolvable sequence of networks.

3.7 Complexity, heuristics and multiple solutions

Since ILP is, in general, NP-hard (Karp, 1972), optimal solutions for very large networks may not be found in polynomial time. Typically, the worst-case time complexity is exponential in the number of variables. In our formulation, for an extant network of n nodes, there are $O(n^2)$ node indicator variables, $O(n^3)$ edge indicator variables and $O(n^3)$ indicator variables for phantom edges.

For many practical ILP problems, it is infeasible to explore the entire search space of solutions. A common approach is to use Linear Programming (LP) relaxations (where the variables are not constrained to be integers) which can be solved efficiently in polynomial time. The objective function value of the LP relaxation provides a theoretical upper bound (for maximization problems) on the ILP solution. This fact is used in branch-and-bound methods to divide the problem in to smaller sub-problems in a hierarchical manner and determine which sub-problems can be eliminated from the

search. Cutting-plane methods also uses LP relaxations to reduce the search space. Known feasible solutions from other heuristics (e.g. in our case the solution from ReverseDMC) can also be used to reduce the search space. During such explorations, multiple sub-optimal but feasible solutions may be found that are themselves useful. See Smith and Taskin (2008) for a recent tutorial focusing on biomedical applications and Wolsey (2020) for more details.

The development of heuristics for solving ILP is an active research area; due to its practical importance efficient software implementations are also being developed and updated regularly. These solvers have various heuristics to limit the size of the branch-and-bound tree and reduce the search space. These heuristics are designed to run for a pre-specified period of time, during which they find multiple (near-optimal) solutions. Although in practice, there is a trade-off between the running time and quality of solutions found, there are no formal guarantees, in general, of quality improvement over time.

In our case, multiple histories provided by the ILP solver yield different evolutionary histories. They may be of equal likelihood and is often the case in our experiments. Even when the histories have the same likelihood, they can differ in other characteristics, such as node arrival order or network structures. We illustrate this with a simulation in Supplementary Appendix S5. Thus, while likelihood can be a good first criterion to select a reconstructed history, multiple solutions should further be evaluated in the light of other biological evidence to select the most plausible history. We discuss this further in the following sections, where small networks allow us to investigate the entire reconstructed history manually.

3.8 Designing heuristics using ILP

The time and memory requirements of the ILP heuristics from standard solvers may be prohibitively expensive for large networks. Supplementary Appendices S6.4 and S7.1 discuss these performance issues further. Nevertheless, ILP solvers can also be useful in subroutines within larger heuristics to solve the problem. To illustrate this, we design a simple Divide and Conquer Heuristic using the ILP, called DCH-ILP. In DCH-ILP, the input network is first partitioned into smaller subgraphs. The ILP is used to reconstruct history of each subgraph. Finally, the reconstructed histories of each subgraph are combined to form the final reconstructed history. There are many ways to design the partitioning step and the final combination step. As a preliminary study we choose simple design choices for these steps as detailed in Supplementary Appendix S7.

4 Experiments

4.1 Simulations

We test the performance of our algorithm and ReverseDMC, the Greedy approach of Navlakha and Kingsford (2011), in simulations where the complete evolutionary history is known. Evolution is simulated following the DMC model starting from an initial network of two connected nodes. For each simulated instance, the extant network is provided as input to ReverseDMC and ILP. The maximum runtime of ILP is fixed to 24 h for each instance. All experiments were run on a server running Ubuntu 16.04.6 with 50 cores [Intel(R) Xeon(R) Gold 6130 CPU 2.10 GHz], and 1 TB RAM. Gurobi (version 7.5.2) (www.gurobi.com), a state-of-the-art solver which is free for academic use, was used to solve the ILP in all the experiments.

We use four evaluation metrics to assess the reconstructed histories. The likelihood of the entire reconstruction is our first metric. The second metric evaluates the node arrival order during evolution. For the inferred histories this is determined by inverting the list of *removed nodes* from each step of the reconstruction. The correlation between this ranked list and the true arrival order are compared using Kendall's Tau (Kendall, 1945) (definition given in Supplementary Appendix S4). Our third metric compares the networks generated at each step of reconstruction with their corresponding true networks. We use graph kernels, which provides a measure of structural similarity between graphs (Vishwanathan *et al.*, 2010). For an extant network G_t, given the true sequence of graphs, $G_S = G_3, \ldots, G_{t-1}$ and a reconstructed history, $\hat{G}_S = \hat{G}_3, \ldots, \hat{G}_{t-1}$, we compute the kernel similarity: $\sum_{i=3}^{t} k(G_i, \hat{G}_i)/(t-2)$, where k is a graph kernel. This measures the average similarity of the entire reconstruction. We use the Weisfeiler-Lehman kernel that measures similarity based on isomorphism of subgraphs within the input graphs (Shervashidze *et al.*, 2011). In these three metrics, higher values indicate better performance.

Due to the symmetry between the anchor and duplicated nodes in the DMC model, at every evolutionary step, switching the new node and the anchor does not change the likelihood. This makes Kendall's Tau a difficult performance metric: a high score is hard to obtain for any algorithm based on DMC model. So, we use a fourth metric, the Robinson-Foulds (RF) distance (Robinson and Foulds, 1981) between binary duplication history trees. A duplication history tree is constructed from a given history (true or inferred) in the following top-down manner. The first node, in G_1, forms the root and the nodes in G_2 are attached as children to the root. At each step of the evolutionary history the anchor node in G_{g-1} becomes the parent to the pair of duplicated nodes in G_g, that are created and attached to the parent in the tree. At the end of this process, we obtain a tree where the leaves correspond to the nodes in the extant network and the internal nodes represent duplication events, without distinguishing the newly duplicated node and its anchor. We calculate the RF distance between duplication trees from a reconstructed history and the true evolutionary history. Lower RF distance indicates better performance.

4.1.1 Reconstruction with true model parameters

We simulated 1500 extant networks with number of nodes in the extant network varying from 6 to 10. For each simulation, the value of each DMC parameter (q_{con} and q_{mod}) was randomly chosen from the interval $[0.1, 0.9]$, rounded to one decimal. The same parameters were used during reconstruction, for both ReverseDMC and our ILP.

Figure 7 shows that there were no simulations where solutions from ILP have a lower likelihood than that of ReverseDMC. Since these are small networks, both ReverseDMC and ILP were able to find optimal solutions in 76% of the cases, while in 24% of the cases ILP found solutions with higher likelihood. The fact that ILP could find solutions with higher likelihood shows that ReverseDMC is not guaranteed to find optimal solutions. Histories reconstructed from ILP had better correlation with the true histories, with respect to node arrival order, in 93% of the cases. With respect to RF distance, 76% of the histories, from ILP and ReverseDMC, had equal RF distance to the true evolutionary tree. In 20% of the cases trees from ILP reconstructions were closer to the true tree, while trees from

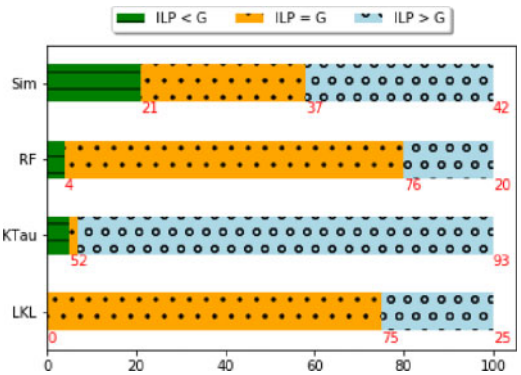

Fig. 7. Reconstruction with true model parameters. Proportion of simulations where reconstructed histories from ILP scored better (ILP > G), equal (ILP=G) and worse (ILP < G) than the reconstructions from ReverseDMC, the Greedy approach of Navlakha and Kingsford (2011), for four metrics - log likelihood (LKL), Kendall's Tau (KTau), RF Distance (RF) and Kernel Similarity (Sim). Numbers below each bar indicate percentages

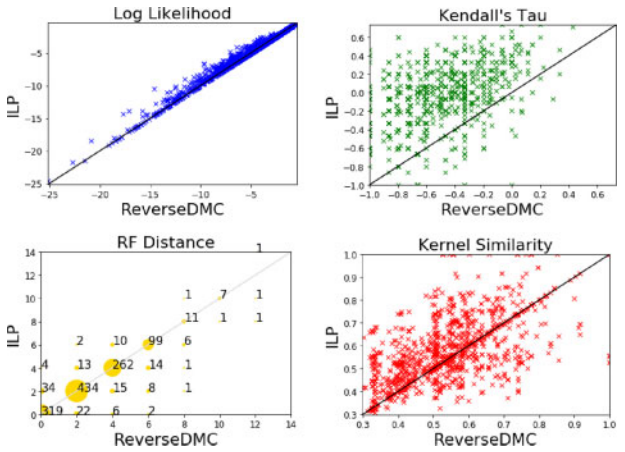

Fig. 8. Scatterplot of log likelihood (top left), Kendall's Tau (top right), RF Distance (bottom left) and Kernel Similarity (bottom right) values from reconstructions from ILP (y axis) and ReverseDMC (x axis). Each marker corresponds to reconstruction from a single extant network, except for RF distance where the size of the marker is proportional to the number of extant networks, shown in the annotated text, for which the pair of values are obtained. For log likelihood, Kendall's Tau and Kernel Similarity, instances above the diagonal indicate better performance scores by ILP. For RF distance instances below the diagonal indicate better performance scores by ILP. In all plots, diagonal indicates equal performance scores

ReverseDMC reconstructions were closer to the true tree only in 4% of the cases. The kernel similarity values of the reconstructed networks were not lower than those from ReverseDMC in 79% of the cases.

Each subplot in Figure 8 plots 1500 pairs of performance scores $(S_{\text{ReverseDMC}}, S_{\text{ILP}})$, one from each simulation. The diagonal in each subplot indicates equal scores from both ILP and ReverseDMC. The scatterplot of log likelihood shows that all the ILP log likelihood scores are either higher (above the diagonal) or equal (on the diagonal) to that of ReverseDMC. The increase in log likelihood is not much since these are small extant networks. Kendall's Tau and Kernel Similarity scores of most of the simulations are higher for ILP and the increase in Kendall's Tau can be considerably high for many instances. RF distances are discrete and we again see that there can be up to 4 units of improvement even in these small trees.

Overall, reconstructed histories from ILP have higher likelihood and obtain node arrival orders, duplication events and inferred networks that are closer to the true evolutionary history compared to those from ReverseDMC.

4.1.2 Additional simulations

We evaluate two other settings:

- Reconstruction from a noisy extant network using the true model parameters.
- Reconstruction from the extant network when the true model parameters are unknown.

The details of these simulations can be found in Supplementary Appendices S6.1 and S6.2. The relative performance trend remains the same as before: reconstructed histories from ILP have higher likelihood and obtain node arrival orders, duplication events and inferred networks that are closer to the true evolutionary history compared to those from ReverseDMC. With increasing noisy edges (up to 20%) in the extant networks there is no significant change in the proportion of solutions where ILP obtains better, equal or worse solutions compared to ReverseDMC with respect to each of the three metrics. When true model parameters $(q_{\text{mod}}, q_{\text{con}})$ are not used during reconstruction, the performance of both ReverseDMC and ILP deteriorates with increasing deviation from the true parameters; and is more affected by mismatch in q_{mod} values. This is consistent with the findings in Navlakha and Kingsford (2011) for ReverseDMC.

The ILP results above are obtained using the solution from ReverseDMC as a known feasible solution to initialize the ILP solver. We study in more detail, in Supplementary Appendix S6.3, the effect of initializing the ILP with a feasible solution from ReverseDMC in two cases: (i) on small input networks where ILP finds the optimal solution within a few hours, and (ii) on larger input networks, where the ILP cannot find an optimal solution in reasonable time, but ILP heuristics are used to find sub-optimal solutions with a pre-specified running time. We find that the initialization makes no difference to either the running time or the likelihood of the solution obtained.

In Supplementary Appendix S7.1, we evaluate the performance of DCH-ILP on large simulated input networks of sizes up to 400 nodes. We study the effect of pre-specified runtime limits for ILP within DCH-ILP on the quality of the solutions obtained, and find that in general, increasing the runtime leads to improvement in likelihood of the solution.

4.2 Real networks

We reconstruct the history of two PPI networks using both ReverseDMC and ILP algorithms. Each algorithm is run for a range of values of q_{con} and q_{mod} ($q_{\text{con}} \in [0.1, 0.4, 0.7, 0.9]$ and $q_{\text{mod}} \in [0.3, 0.7]$), the solution with the best likelihood is chosen for further analysis. The maximum runtime of ILP is fixed to 24 h for each instance. The solution from ReverseDMC is used as a known feasible solution to initialize the ILP solver. For each extant network we analyze two solutions, denoted by ILP1 and ILP2, among the multiple solutions given by the ILP solver.

Since the true evolutionary histories are not known, we cannot use the evaluation metrics that are used in our simulation studies. 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.*, 2019), 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 Supplementary Appendix S6). 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 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 $\hat{G}_S$. The network $\hat{G}_2$ 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.

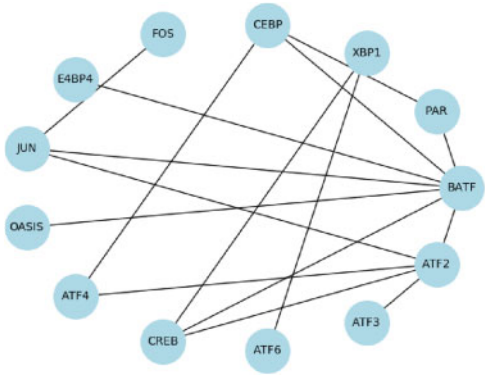

Fig. 9. Extant bZIP network used in our experiment

Table 1. Likelihood, node arrival time accuracy and ADSS for ancestral reconstruction of the bZIP network obtained with $q_{con} = 0.7, q_{mod} = 0.4$

Algorithm	Log-likelihood	Kendall's Tau	ADSS
ReverseDMC	−21	−0.23	−1901.55
ILP1	−19.6	0.18	−1797.11
ILP2	−19.6	−0.03	−1749.31

4.2.1 bZIP transcription factors

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. 9).

Table 1 shows the likelihood, Node Arrival Time Accuracy (measured by Kendall's Tau) and ADSS of the solutions obtained from ReverseDMC and ILP, for ancestral reconstruction of the bZIP Network. With respect to all three metrics, the two solutions obtained by ILP (ILP1 and ILP2) are better than that of ReverseDMC. Between ILP1 and ILP2, ILP1 has better Kendall's Tau and ILP2 has better ADSS. Table 2 shows the order of arrival of proteins of the two ILP solutions and the solution from ReverseDMC. The order based on ortholog information that is used to calculate Kendall's Tau is also shown in the leftmost column.

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 2 we observe that these three proteins appear early in the order of both ILP1 and ILP2 (before the seventh step) while JUN and ATF3 arrive after the seventh step in the order inferred by ReverseDMC. Comparing ILP1 and ILP2, we notice that the major difference between the two solutions lies in the arrival order of ATF6 and XBP1. It is known that XBP1 and ATF6 interact in the pathway of unfolded protein response (UPR) (Mitra and Ryoo, 2019). However, to our knowledge, there is no evidence favoring the earlier arrival of one or the other.

4.2.2 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 (Liebeskind *et al.*, 2019; Mallam and Marcotte, 2017). 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.* (2019), shown in Figure 10, as the extant network for ancestral reconstruction.

Table 3 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 solutions obtained by ILP are better than that of ReverseDMC. Table 4 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 solution ILP1, 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. ILP2, on the other hand, despite having similar values in all the three metrics, to those from ILP1, has COMMD1 in the eigth step of evolution. Thus, in the light of available additional evidence, solution ILP1 should be favored over solution ILP2.

In Supplementary Appendix S7.2, we discuss the reconstruction, from DCH-ILP and ReverseDMC, of a larger Mouse

Table 2. Left: Anchor proteins in the bZIP network given in descending order of the number of orthologs

Gene	Orthologs	Time step	ReverseDMC	ILP1	ILP2
JUN	820	2	FOS,	ATF6,	FOS,
ATF2	546		CREB	ATF2	ATF2
ATF6	453	3	ATF6	FOS	XBP1
ATF4	432	4	BATF	ATF3	ATF3
FOS	339	5	ATF4	CREB	JUN
OASIS	285	6	ATF2	JUN	CREB
CREB	268	7	E4BP4	CEBP	CEBP
CEBP	259	8	JUN	BATF	BATF
E4BP4	255	9	ATF3	PAR	PAR
PAR	244	10	CEBP	ATF4	ATF4
ATF3	229	11	PAR	E4BP4	E4BP4
BATF	227	12	XBP1	OASIS	OASIS
XBP1	203	13	OASIS	XBP1	ATF6

Note: Right: Arrival order at each step of evolution, based on reconstructions from ReverseDMC and two solutions from ILP.

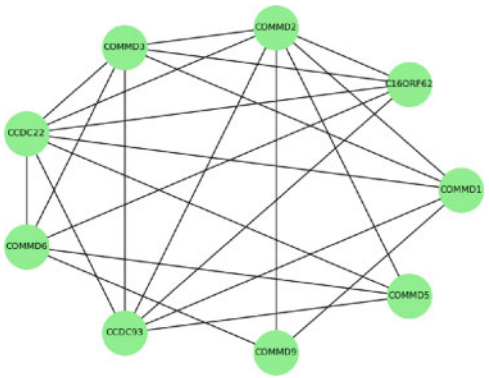

Fig. 10. Extant Commander network used in our experiment

Table 3. Likelihood, node arrival time accuracy and ADSS for ancestral reconstruction of the commander network obtained with $q_{con} = 0.7, q_{mod} = 0.4$

Algorithm	Log-likelihood	Kendall's Tau	ADSS
ReverseDMC	−17.49	−0.27	−2873.25
ILP1	−16.17	0.01	−1649.64
ILP2	−16.17	0.01	−1689.55

Table 4. Left: Anchor proteins in the commander network given in descending order of the number of orthologs

Gene	Orthologs	Time step	Reverse DMC	ILP1	ILP2
ccdc93	508	2	COMMD6,	COMMD3,	COMMD5,
c16orf62	501		COMMD9	COMMD9	COMMD9
commd2	422	3	C16ORF62	COMMD1	COMMD2
ccdc22	414	4	COMMD2	COMMD6	COMMD6
commd5	384	5	CCDC93	COMMD2	CCDC93
commd3	348	6	CCDC22	COMMD5	COMMD3
commd1	290	7	COMMD5	CCDC93	CCDC22
commd9	273	8	COMMD1	CCDC22	COMMD1
commd6	158	9	COMMD3	C16ORF62	C16ORF62

Note: Right: Arrival order of anchor proteins in the commander network, at each step of evolution, based on reconstructions from ReverseDMC and two solutions from ILP.

Cytomegalovirus Herpesvirus PPI network (Fossum *et al.*, 2009) comprising 111 proteins.

5 Conclusion

We presented an ILP formulation for maximum-likelihood reconstruction of the evolution of a PPI network using the DMC model. We proved the correctness of the ILP formulation, that is designed to find the optimal solution. Since ILP is NP-hard, for large networks, an optimal solution may not be found in polynomial time. However, heuristics from general-purpose ILP solvers can be used to find multiple near-optimal solutions in a pre-specified period of time.

We compared the solutions obtained by ILP heuristics 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 superior to those from ReverseDMC, in terms of not only likelihood, but also node arrival order, duplication history and similarity of reconstructed ancestral networks to the true ones. Further, the ILP solutions are robust to noise in extant networks and mismatch in input model parameters. 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. We also illustrate the value of obtaining multiple solutions on both simulated and real data. Even when solutions are of equal likelihood, they differ in other characteristics such as node arrival order or network structure. Our analysis emphasizes the need for considering multiple equi-likelihood histories in the light of additional biological evidence.

A limitation of our solution is the running time of ILP heuristics—it can take a considerably long time to find good solutions for large networks. Memory requirements of ILP solver also increase substantially with increase in size of input networks. However, the ILP can be useful as a sub-routine in larger scalable heuristics. We conducted a preliminary study with DCH-ILP, a simple divide-and-conquer strategy where the input network was partitioned into smaller networks and then solved using our ILP. Despite the strong assumptions made about the partitions, DCH-ILP performs well on the networks tested. Future work can improve the method by investigating other design choices made in the heuristic.

To our knowledge, this is the first ILP to a model-based network reconstruction problem. We believe it is a valuable tool for further theoretical and algorithmic development in network reconstruction problems. For instance, the constructs used to design the ILP can be used for other evolutionary models such as the Preferential Attachment model. The ILP framework could also be generalized to handle multiple input networks as well as to take into account additional information, such as gene duplication histories.

Acknowledgements

Part of the work was performed during X.Z.'s visit at the Simons Institute for the Theory of Computing at University of California, Berkeley. The authors 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 [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. and Z.Z. were supported in part by the US National Science Foundation [DBI-2019771].

Conflict of Interest: C.K. is a co-founder of Ocean Genomics.

References

Amoutzias,G. *et al.* (2006) One billion years of bZIP transcription factor evolution: conservation and change in dimerization and DNA-binding site specificity. *Mol. Biol. Evol.*, **24**, 827–835.

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

Bartuzi,P. *et al.* (2016) CCC- and WASH-mediated endosomal sorting of LDLR is required for normal clearance of circulating LDL. *Nat. Commun.*, **7**, 10961.

Borenstein,E. and Feldman,M.W. (2009) Topological signatures of species interactions in metabolic networks. *J. Comput. Biol.*, **16**, 191–200.

Chung,F. *et al.* (2003) Duplication models for biological networks. *J. Comput. Biol.*, **10**, 677–687.

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

Fong,J.H. *et al.* (2004) Predicting specificity in bZIP coiled-coil protein interactions. *Genome Biol.*, **5**, R11.

Fossum,E. *et al.* (2009) Evolutionarily conserved herpesviral protein interaction networks. *PLoS Pathog.*, **5**, e1000570.

Jasra,A. *et al.* (2015) Bayesian inference for duplication–mutation with complementarity network models. *J. Comput. Biol.*, **22**, 1025–1033.

Jin,Y. *et al.* (2013) The evolutionary dynamics of protein–protein interaction networks inferred from the reconstruction of ancient networks. *PLoS One*, **8**, e58134.

Karp,R.M. (1972) Reducibility among combinatorial problems. In: Miller R.E. *et al.*, (eds.), *Complexity of Computer Computations*. The IBM Research Symposia Series. Springer, Boston, MA, pp. 85–103.

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

Kreimer,A. *et al.* (2008) The evolution of modularity in bacterial metabolic networks. *Proc. Natl. Acad. Sci. USA*, **105**, 6976–6981.

Kriventseva,E.V. *et al.* (2019) OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Res.*, **47**, D807–D811.

Leskovec,J. *et al.* (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In Proceedings of the 11th ACM SIGKDD

International Conference on Knowledge Discovery and Data Mining. ACM, pp. 177–187.
Li,S. *et al.* (2013) Maximum likelihood inference of the evolutionary history of a PPI network from the duplication history of its proteins. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **10**, 1412–1421.
Liebeskind,B.J. *et al.* (2019) Ancestral reconstruction of protein interaction networks. *PLoS Comput. Biol.*, **15**, e1007396.
Mallam,A.L. and Marcotte,E.M. (2017) Systems-wide studies uncover commander, a multiprotein complex essential to human development. *Cell Syst.*, **4**, 483–494.
Middendorf,M. *et al.* (2005) Inferring network mechanisms: the *Drosophila melanogaster* protein interaction network. *Proc. Natl. Acad. Sci. USA*, **102**, 3192–3197.
Mitra,S. and Ryoo,H.D. (2019) The unfolded protein response in metazoan development. *J. Cell. Sci.*, **132,** jcs217216.
Navlakha,S. and Kingsford,C. (2011) Network archaeology: uncovering ancient networks from present-day interactions. *PLoS Comput. Biol.*, **7**, 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. *J. Mol. Biol.*, **48**, 443–453.
Nichols,S.A. *et al.* (2006) Early evolution of animal cell signaling and adhesion genes. *Proc. Natl. Acad. Sci. USA*, **103**, 12451–12456.
Patro,R. and Kingsford,C. (2013) Predicting protein interactions via parsimonious network history inference. *Bioinformatics*, **29**, i237–i246.
Patro,R. *et al.* (2012) Parsimonious reconstruction of network evolution. *Algorithms Mol. Biol.*, 7, 25.
Pereira-Leal,J.B. *et al.* (2006) The origins and evolution of functional modules: lessons from protein complexes. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, **361**, 507–517.
Pinney,J.W. *et al.* (2007) Reconstruction of ancestral protein interaction networks for the bZIP transcription factors. *Proc. Natl. Acad. Sci. USA*, **104**, 20449–20453.
Riera-Romo,M. (2018) COMMD1: a multifunctional regulatory protein. *J. Cell. Biochem.*, **119**, 34–51.
Robinson,D.F. and Foulds,L.R. (1981) Comparison of phylogenetic trees. *Math. Biosci.*, **53**, 131–147.
Shervashidze,N. *et al.* (2011) Weisfeiler–Lehman Graph Kernels. *J. Mach. Learn. Res.*, **12**, 2539–2561.
Smith,J.C. and Taskin,Z.C. (2008) A tutorial guide to mixed-integer programming models and solution techniques. In: Lim, G. and Lee, E. (eds.) *Optimization in Medicine and Biology*. Auerbach Publications, New York. pp. 521–548.
Sreedharan,J.K. *et al.* (2019) Inferring temporal information from a snapshot of a dynamic network. *Sci. Rep.*, **9**, 3057.
Vázquez,A. *et al.* (2003) Modeling of protein interaction networks. *Complexus*, **1**, 38–44.
Vidal,M. *et al.* (2011) Interactome networks and human disease. *Cell*, **144**, 986–998.
Vishwanathan,S.V.N. *et al.* (2010) Graph kernels. *J. Mach. Learn. Res.*, **11**, 1201–1242. [CVOCROSSCVO]
Wagner,A. (2001) The yeast protein interaction network evolves rapidly and contains few redundant duplicate genes. *Mol. Biol. Evol.*, **18**, 1283–1292.
Wolsey,L.A. (2020). Integer programming. John Wiley & Sons.
Yamada,T. and Bork,P. (2009) Evolution of biomolecular networks – lessons from metabolic and protein interactions. *Nat. Rev. Mol. Cell Biol.*, **10**, 791–803.
Young,J.-G. *et al.* (2019) Phase transition in the recoverability of network history. *Phys. Rev. X*, **9**, 041056.
Zhang,J. *et al.* (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.E. (2012) Refining regulatory networks through phylogenetic transfer of information. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **9**, 1032–1045.