BIOINFORMATICS

Vol. 29 ISMB/ECCB 2013, pages i237–i246
doi:10.1093/bioinformatics/btt224

Predicting protein interactions via parsimonious network history inference

Rob Patro* and Carl Kingsford*
Lane Center for Computational Biology, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

ABSTRACT

Motivation: Reconstruction of the network-level evolutionary history of protein–protein interactions provides a principled way to relate interactions in several present-day networks. Here, we present a general framework for inferring such histories and demonstrate how it can be used to determine what interactions existed in the ancestral networks, which present-day interactions we might expect to exist based on evolutionary evidence and what information extant networks contain about the order of ancestral protein duplications.

Results: Our framework characterizes the space of likely parsimonious network histories. It results in a structure that can be used to find probabilities for a number of events associated with the histories. The framework is based on a directed hypergraph formulation of dynamic programming that we extend to enumerate many optimal and near-optimal solutions. The algorithm is applied to reconstructing ancestral interactions among bZIP transcription factors, imputing missing present-day interactions among the bZIPs and among proteins from five herpes viruses, and determining relative protein duplication order in the bZIP family. Our approach more accurately reconstructs ancestral interactions than existing approaches. In cross-validation tests, we find that our approach ranks the majority of the left-out present-day interactions among the top 2 and 17% of possible edges for the bZIP and herpes networks, respectively, making it a competitive approach for edge imputation. It also estimates relative bZIP protein duplication orders, using only interaction data and phylogenetic tree topology, which are significantly correlated with sequence-based estimates.

Availability: The algorithm is implemented in C++, is open source and is available at http://www.cs.cmu.edu/ckingsf/software/parana2.

Contact: robp@cs.cmu.edu or carlk@cs.cmu.edu

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

1 INTRODUCTION

Improved techniques for understanding how collections of protein interactions have evolved over time have a number of applications. For example, they can help identify stable and rewired modules (Kreimer *et al*., 2008) and protein complexes (Pereira-Leal *et al*., 2007). The quality of inferred networks under various parameters can help estimate the probabilities of different evolutionary events (Li *et al*., 2012; Middendorf *et al*., 2005; Navlakha and Kingsford, 2011) or correct for phylogenetic branch lengths (Zhu and Nakhleh, 2012). Ancestral network reconstruction has been explored to improve network alignment algorithms (Dutkowski and Tiuryn, 2007; Flannick *et al*., 2006, 2009; Singh *et al*., 2007). The study of ancestral metabolic pathways can reveal how changes in metabolic pathways relate to changes in the environment (Borenstein and Feldman, 2009; Borenstein *et al*., 2008; Mithani *et al*., 2009). Zhang and Moret (2008, 2010) apply network evolution inference to improve inference of regulatory networks in present-day species.

Previous algorithms for network history reconstruction include the use of graphical models (Dutkowski and Tiuryn, 2007; Pinney *et al*., 2007), greedy local search (Navlakha and Kingsford, 2011) and extensions thereof (Li *et al*., 2012; Zhu and Nakhleh, 2012), maximum-likelihood inference (Zhang and Moret, 2008, 2010) and other approaches (Gibson and Goldberg, 2009). Patro *et al*. (2012) introduced a new parsimony framework that modeled the problem as one of finding the fewest number of interaction gain and loss events that reconstruct the observed present-day networks. Many of these previous approaches find only one possible network history and make inferences based on that single history. However, there may be a large number of optimal and near-optimal histories. A priori, we do not know how different these solutions may be, or how representative of the ensemble the solution at which we arrive is. Further, although maximum-likelihood–based approaches do not necessarily produce a single history, Carvalho and Lawrence (2008) suggest that such estimators may not generally characterize the posterior-weighted ensemble of solutions well.

A maximum-likelihood network history inference method has been applied to the problem of predicting regulatory interactions in present-day networks (Zhang and Moret, 2008, 2010). However, that approach requires a known complete ordering of the duplication events in each homology group, which our approach does not. Further, being based on a parameterized network evolution model, it requires the estimation of numerous model parameters.

To overcome these limitations, we present an approach, based on a novel algorithm and advanced dynamic programming techniques, which is able to efficiently characterize the relevant portion of the space of network histories without resorting to sampling. By formulating our dynamic program in the forward hypergraph framework (Gallo *et al*., 1993), it becomes clear how to explore the space of solutions. We develop an extension of the *k*-best parsing algorithm of Huang and Chiang (2005) that allows us to aggregate solutions of equivalent quality. As a result, rather than enumerating individual solutions, we are able to enumerate solution classes (i.e. the set of all solutions having the same cost) and provide a characterization of the space of optimal and near-optimal solutions to an instance of the network history inference problem. Inspired by Feynman and Brown (1942), we call this method a sum-over-parsimonious-histories (SOPH) approach to

*To whom correspondence should be addressed.

ancestral network reconstruction. Although related to certain approaches in natural language processing, our approach for generating a weighted ensemble of parsimonious solutions is novel and may also prove useful in other areas of computational biology.

For every potential interaction—either ancestral or extant—our algorithm computes the posterior probability, summed over an ensemble of parsimonious and near-parsimonious histories, which the interaction exists. We show this approach outperforms the graphical model formulation used in Pinney *et al.* (2007) and Dutkowski and Tiuryn (2007). Further, as posterior probabilities are provided for all potential interactions (including extant ones) that participate in the ensemble, we are able to impute missing interactions and to quantify the consistency—in terms of evolutionary parsimony—of a given set of interactions.

When applied to the problem of predicting ancestral interactions among bZIPs, the SOPH approach is particularly beneficial when noise is added to the present-day networks. These noisy networks simulate the common scenario in which the measurement of present-day interactions is error-prone. The SOPH method seems to be both accurate and robust. Further, anecdotally, Fossum *et al.* (2009) argue that the interaction between KSHV-1 proteins UL33 and UL31 is highly conserved across many herpes species, and we find that our SOPH approach predicts an ancestral interaction between the orthology groups of these proteins with the second highest probability among all potential ancestral interactions.

We test the approach's ability to predict missing edges in present-day networks, and we show that it often outperforms a state-of-the-art approach for edge prediction based on network topology (Lei and Ruan, 2013). On the bZIP transcription factor network, where we perform leave-one-out, 5-fold and 10-fold cross-validation, we find that our approach most often puts edges from the test set in the top 1% of the probabilities assigned to pairs.

We also perform edge prediction on a collection of five herpes virus protein interaction networks (Fossum *et al.*, 2009) in a similar leave-one-out setting (the data are too sparse for higher-fold cross-validation). Here, the left-out edge is, on average, in the top 25% of high-probability edges. We also breakdown performance based on which orthology groups the interaction participants are members of, and find that the good performance is driven by generally good performance for most pairs of orthology groups. As these data are believed to have high–false-negative rate, there surely are real missing edges in the given present-day networks, meaning the actual performance is likely in fact better.

The ensemble of parsimonious network histories encoded by our framework can be used to answer other types of queries about the network histories that are not even possible in existing maximum-likelihood approaches that work based on interaction trees. As an illustration, we use the SOPH framework to predict the relative duplication order between pairs of ancestral bZIP proteins. Existing maximum-likelihood approaches (Dutkowski and Tiuryn, 2007; Pinney *et al.*, 2007; Zhang and Moret, 2008, 2010) cannot perform this task, as a total order of duplication events is required for the inference procedure used by those algorithms. We find that the relative duplication orders predicted by our SOPH framework, which were predicted without the use of phylogenetic branch length information, are significantly correlated with the duplication order derived from the protein sequences.

2 APPROACH

2.1 Overview

At a high level, we formulate the network history inference problem as a combinatorial optimization problem that seeks a parsimonious or low-cost set of interaction gain and loss events that explain the observed present-day networks. We rewrite the combinatorial problem by encoding it as an instance of the *optimal derivation* problem on a directed ordered hypergraph that allows us to efficiently count the number of solutions of various costs that are close to the optimal and to compute the probability that any particular interaction gain or loss event is present in the ensemble of near-optimal histories. The ensemble of histories that is compactly encoded by the hypergraph can also be used to answer other queries about the histories themselves, such as inferring the relative duplication order of proteins within a species.

2.2 The network history inference problem

The network history inference problem seeks to find a set of gains and losses of protein interactions that is consistent with both the observed present-day interactions and the phylogenetic history relating the proteins. Formally, we are given present-day networks $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ for species $S = \{1, \ldots, k\}$. We are also given a set $\mathcal{T}$ of binary phylogenetic trees where $T \in \mathcal{T}$ has leaves associated with a subset of $\mathcal{V} = \bigcup_i V_i$. Every $v \in \mathcal{V}$ appears as a leaf in at most one tree, and without loss of generality, we may assume that every $v \in \mathcal{V}$ appears in exactly one such tree. Nodes in each tree are labeled as either protein duplication events or speciation events.

An *interaction event* is a triple (u, v, a), where $u \in T_1 \in \mathcal{T}$, $v \in T_2 \in \mathcal{T}$ and $a \in \{\mathbf{gain}, \mathbf{loss}\}$. T_1 may equal T_2 but neither u nor v can be an ancestor of the other. If $a = \mathbf{gain}$, the event represents the gain of an interaction between the ancestral proteins u and v. If $a = \mathbf{loss}$, it represents the loss of an interaction.

Interactions are assumed to be inherited through duplication events. An interaction exists between two proteins if it has been gained between a pair of their ancestors and not subsequently lost. Specifically, given a set $\mathcal{I}$ of interaction events, an interaction exists between u and v if there are ancestors x of u and y of v such that the event $(x, y, \mathbf{gain})$ is in $\mathcal{I}$, and there are no nodes x', y' such x' is an ancestor of u and a descendant of x, and y' is an ancestor of u and descendent of y such that $(x', y', \mathbf{loss})$ is in $\mathcal{I}$.

We say that a set $\mathcal{I}$ of interaction events

- is *valid* if the events are logically and temporally consistent. That is, a gain event occurs only at a time when the edge does not exist, a loss event occurs only when the edge exists and time ranges can be assigned to every node such that events only happen between pairs of nodes that have overlapping time ranges (note that we do not explicitly find these time ranges).

- *reconstructs* $G_1, \ldots, G_k$ if, for all $u, v \in \mathcal{V}$, $\mathcal{I}$ implies that edge $\{u, v\}$ exists if and only if that edge is present in $G_1, \ldots, G_k$.

The network history inference problem is then

Problem 1. (Network History Inference) *Find the smallest set $\mathcal{I}$ of triples on $\mathcal{T}$ that represents a valid history of $G_1, \ldots, G_k$ and that reconstructs the present-day networks $G_1, \ldots, G_k$. If a function $c(e)$ that assigns a cost to interaction event e is given, we seek the lowest-cost set $\mathcal{I}$.*

Finding a score-weighted ensemble of solutions to this problem allows us to solve the related problems of (i) predicting ancestral interaction networks; (ii) imputing missing interactions in present-day networks; and (iii) inferring relative orders for duplication events that are consistent with a molecular clock.

3 METHODS

3.1 Directed ordered hypergraphs

We use the hypergraph definition and a number of related definitions given by Huang and Chiang (2005). Specifically, we define a directed ordered hypergraph as $H = (V_H, E_H, r, c)$, where V is the set of vertices, E is the set of ordered hyperarcs, $r \in V$ is a designated root node and $c : E_H \rightarrow \mathbb{R}$ is a function assigning costs to the hyperedges. Each hyperarch e is a pair $(\mathbf{h}(e), \mathbf{t}(e))$, where $\mathbf{h}(e)$ is a vertex called the *head* of the hyperarc and $\mathbf{t}(e)$ is an ordered list of vertices called the *tail* of the hyperarc. We denote by $\mathbf{t}_i(e)i$ the ith element of the tail of e. Without loss of generality, we will assume that every vertex is the head of some hyperarc e; the tail of e can be an empty list (denoted here as $\langle\rangle$). A hyperarc with head x and tail $y_1, \ldots$ is written as $x \leftarrow \langle y_1, \ldots \rangle$.

We call the set of hyperarcs with v as their head the *backward star* of v, and denote it by $\mathrm{BS}(v) = \{e \in E_H | v = \mathbf{h}(e)\}$. Any vertex w that appears in the tail of some hyperarc e where $e \in \mathrm{BS}(v)$ is said to precede v.

3.2 The optimal derivation problem

We will formulate the network history inference problem as an instance of the optimal derivation problem in the ordered hypergraph framework (Huang and Chiang, 2005). This framework allows one to explicitly represent the space of solutions to certain classes of combinatorial problems by encoding these solutions in the topology of a directed ordered hypergraph. Such an ordered hypergraph representation is used in a wide variety of different fields, including natural language processing (Huang and Chiang, 2005; Klein and Manning, 2001) and operations research (Nielsen *et al.*, 2005). The highly similar directed hypergraph framework was first introduced in computational biology by Finkelstein and Roytberg (1993), where it was shown how many classical dynamic programming problems from sequence alignment to RNA secondary structure prediction could be formulated in this framework. Recently, Ponty and Saule (2011) applied dynamic programming in the directed hypergraph framework to the problem of pseudoknotted RNA folding, and they extended the algorithm to allow the computation of the moments of additive features (e.g. free-energy and helicies).

The *optimal derivation* of an acyclic, directed, ordered hypergraph is $D^*(r)$, defined recursively by

$$D^*(u) := \min_{e \in \mathrm{BS}(u)} \left\{ c(e) + \sum_i D^*(\mathbf{t}_i(e)) \right\}. \tag{1}$$

By traversing the hypergraph in topological order starting with the nodes that have only zero-length tails, the solutions to subproblems are

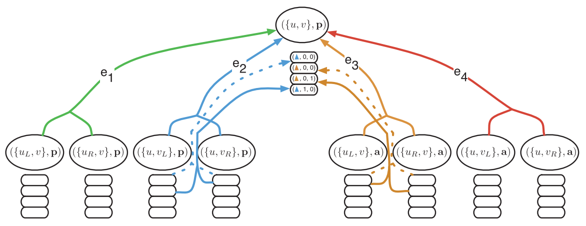

Fig. 1. Mapping recurrence to a hypergraph. An illustration representing a particular recurrence term in the hypergraph. Each hyperarc encodes a set of subterms that must be evaluated to provide a solution to the head vertex $(\{u, v\}, p)$. The arrows denote derivations with back-pointers, and they show the first (dashed blue), second (dashed orange), third (solid blue) and fourth (solid orange) best derivations of the head vertex, and which derivations of the tail vertices were used to achieve them

available when needed. This is the basic strategy behind traditional dynamic programming approaches, and the hypergraph representation simply makes the relation between the terms of the recurrence explicit by encoding them in the topological structure of the hypergraph. Each vertex in the hypergraph represents a term of the recurrence, and the hyperarcs encode the sub-terms (tail nodes of the arc) on which a term (head node of the arc) depends (as illustrated, e.g. in Fig. 1).

3.3 Network history inference as optimal derivation

The network history inference problem can be encoded as an instance of the optimal derivation problem as follows. We set the hypervertices of the hypergraph H to be

$$V_H := \left\{ (\{u, v\}, s) | u, v \in \mathcal{T} \text{ and } s \in \{\textbf{present}, \textbf{absent}\} \right\}. \tag{2}$$

Node $(\{u, v\}, s)$ in H represents whether there is an interaction between proteins u and v just before either of the proteins duplicate. We exclude from V_H any hypervertices involving proteins u, v that cannot have an interaction between them because one is an ancestor of the other or because they are in different species.

Let u_L and u_R denote the left and right children of node u. For every hypernode $(\{u, v\}, \textbf{present})$ where u and v are not leaves, we have the following hyperarcs:

$$(\{u, v\}, \textbf{present}) \leftarrow \langle (\{u_L, v\}, \textbf{present}), (\{u_R, v\}, \textbf{present}) \rangle \tag{3}$$

$$(\{u, v\}, \textbf{present}) \leftarrow \langle (\{u_L, v\}, \textbf{absent}), (\{u_R, v\}, \textbf{absent}) \rangle \tag{4}$$

$$(\{u, v\}, \textbf{present}) \leftarrow \langle (\{u, v_L\}, \textbf{present}), (\{u, v_R\}, \textbf{present}) \rangle \tag{5}$$

$$(\{u, v\}, \textbf{present}) \leftarrow \langle (\{u, v_L\}, \textbf{absent}), (\{u, v_R\}, \textbf{absent}) \rangle \tag{6}$$

The hyperarcs aforementioned encode the option of recursing into either the children of u or the children of v and the option of losing the u–v interaction [Equations (4) and (6)] or not losing it [Equations (3) and (5)]. The cost of hyperarcs (4) and (6) is the cost of a loss event, and the cost of hyperarcs (3) and (5) is 0. The analogous hyperarcs exist for head nodes of the form $(u, v, \textbf{absent})$, with **present** and **absent** switched. Finally, for those hypervertices where $u = v$ (representing potential homodimer interactions), the incoming hyperarcs are slightly different. Specifically, denoting **present** as **p** and **absent** as **a**, a hypervertex $(\{u, u\}, \mathbf{p})$ appears as the head of the following hyperarcs:

$$(\{u, u\}, \mathbf{p}) \leftarrow \langle (\{u_L, u_L\}, \mathbf{p}), (\{u_R, u_R\}, \mathbf{p}), (\{u_L, u_R\}, \mathbf{p}) \rangle \tag{7}$$

$$(\{u, u\}, \mathbf{p}) \leftarrow \langle (\{u_L, u_L\}, \mathbf{a}), (\{u_R, u_R\}, \mathbf{a}), (\{u_L, u_R\}, \mathbf{a}) \rangle. \tag{8}$$

The hyperarc in (7) encodes the recurrence where a homodimer interaction for protein u is inherited by its progeny, implying the edges $\{u_L, u_L\}$, $\{u_R, u_R\}$ and $\{u_L, u_R\}$. The hyperarc in (8) encodes the recurrence in which the homodimer interaction is lost before u's duplication. Just as with Equations (3–6), the analogous hyperarcs exist for hypervertices of the form $(\{u, u\}, \mathbf{a})$ with $\mathbf{p}$ and $\mathbf{a}$ switched.

If u or v is a leaf, we omit the hyperarcs above that would involve u or v's non-existent children. If both u and v are leaves, we add the trivial hyperarcs $(\{u, v\}, \textbf{present}) \leftarrow \langle\rangle$ and $(\{u, v\}, \textbf{absent}) \leftarrow \langle\rangle$ with an empty tail. In this case, the cost of $(\{u, v\}, \textbf{present}) \leftarrow \langle\rangle$ is the cost of a loss event if edge $\{u, v\}$ exists in the observed present-day networks and 0 otherwise; the cost of $(\{u, v\}, \textbf{absent}) \leftarrow \langle\rangle$ is the cost of an interaction gain if present-day edge $\{u, v\}$ exists. We can also assign equal, non-zero costs to leaf nodes to designate that the state of an interaction is unknown rather than present or absent.

THEOREM 2. Let D^* be the set of hyperarcs used in an optimal derivation of the hypergraph defined earlier in the text. Let set $\mathcal{I}$ contain a gain event corresponding to every hyperarc in D^* that transitioned from **absent** to **present** and let $\mathcal{I}$ further contain a loss event corresponding to every hyperarc that transitioned from **present** to **absent**. Then $\mathcal{I}$ is the lowest-cost solution to the network history inference problem (Problem 1), except that $\mathcal{I}$ may contain temporally inconsistent events.

We omit the proof of Theorem 2 because of space, but it follows directly from the proof in Patro *et al.* (2012), translated into the hypergraph framework. The issue of allowing temporally inconsistent events is apparently what makes Problem 1 difficult. Here, we hope to mitigate the effect of temporally inconsistent solutions by summing over many near-optimal solutions.

In the rest of this article, we will refer to solutions having minimum cost as *optimal*, regardless of their inclusion of temporally inconsistent events. Thus, when we say a solution is optimal, we mean that it has the absolute minimum cost with regard to the parsimony criteria of any history generating the extant interactions. This is justified as, in practice, such temporally inconsistent optimal solutions seem to be rare (Patro *et al.*, 2012). When we say that a solution is near-optimal, we mean that it is optimal or it has a cost close to that of an optimal solution; it need not be temporally consistent.

All of the extensions to the recurrence and cost function described in Patro *et al.* (2012) can be encoded in the hypergraph framework, including directed edges, asymmetric interaction gain and loss costs and weighted branch length costs.

3.4 Counting optimal and near-optimal solutions

There may be many near-optimal derivations representing different network histories. We would like to use all these histories to compute probabilities for particular events (e.g. interaction events or an order of duplication events) to have occurred. First, we show how to compute the number of derivations of various costs.

Let $\mathcal{D}_j(x)$ be the set of the jth-best derivations rooted at hypervertex x. That is, $\mathcal{D}_0(x)$ is the set of optimal derivations, and $\mathcal{D}_1(x)$ is the set of non-optimal derivations with cost as close to optimal as possible. We call $\mathcal{D}_j(x)$ a *cost class*, and let $C_j(x)$ denote the cost of each derivation in $\mathcal{D}_j(x)$.

We want to accumulate the sizes of the top-k cost classes of the root of the hypergraph. This will give us a distribution of the costs of near-optimal derivations; thus, a distribution of costs of near-optimal network histories. That is, we would like to compute $|\mathcal{D}_j(x)|$ for $j = 1, \ldots, k$ for some k. In general, this constitutes many more than the top-k individual solutions because there are many ways to obtain different solutions of equivalent cost. The key to developing an efficient algorithm for this task is to realize that we can count all derivations belonging to the top-k cost classes of a vertex without enumerating them.

Every derivation D in $\mathcal{D}_j(x)$ is built up from a choice of hyperarc $e = x \leftarrow \langle t_1, \ldots\rangle$ combined with (potentially near-optimal) choices of derivations of each of the members of the tail $\langle t_1, \ldots\rangle$ of that hyperarc. Derivation D thus includes some subderivations $D_{t_i} \in \mathcal{D}_{d_i}(t_i)$, where d_i is the index of the cost class used in the subderivation for t_i for derivation D. However, the size $|\mathcal{D}_j(x)|$ does not depend on the specific choices of D_{t_i} but only their cost classes d_i. Specifically, a particular choice of hyperarc $e = x \leftarrow \langle t1, \ldots, t_{|e|}\rangle$ and of a set of $\{d_i\}_i$ leads to

$$\#(x \leftarrow \langle t1, \ldots, t_{|e|}\rangle, d_1, \ldots, d_{|e|}) := \prod_i |\mathcal{D}_{d_i}(t_i)| \tag{9}$$

possible derivations of the same cost $c\big(e, d_1, \ldots, d_{|e|}\big)$. Let $\vec{d}$ represent a vector of choices of cost classes [e.g. $\vec{d} = (d_1, \ldots, d_{|e|})$]. Then the size of a cost class can be expressed recursively by combining Equation (9) with

$$|\mathcal{D}_j(x)| = \sum_{\substack{e \in \mathrm{BS}(x) \\ \vec{d}\,:\,c(e,\vec{d}) = C_j(x)}} \#(e, \vec{d}). \tag{10}$$

Unfortunately, implementing the aforementioned sum directly would be computationally expensive, as it involves summing over many choices of $\vec{d}$. However, we can exploit the fact that derivations in cost class $j+1$ are related to derivations in cost class j in the following way. Denote by b_ℓ the vector having a 1 in its ℓth position and a 0 everywhere else. Then, we define the *neighborhood* of a pair $(e, \vec{d})$ to be the set $\mathcal{N}(e, \vec{d}) = \{(e, \vec{d} + b_\ell)\}_{\ell=1}^{|\mathbf{e}|}$. In other words, $\mathcal{N}(e, \vec{d})$ is the set of choices for cost classes for the subderivations that use the same cost classes as $\vec{d}$ except for one item in the tail of e, for which the next higher cost class is used. We then have the following lemma.

LEMMA 3. Let $(e, \vec{d})$ be a derivation that falls in cost class $\mathcal{D}_j(x)$. Then any derivation in $\mathcal{D}_{j+1}(x)$ is in $\mathcal{N}(e, \vec{d})$.

Again, for space, we omit a full proof, but the lemma is intuitive: to go up one cost class you should only change the cost class used for one of the subderivations. This is the essential observation behind so-called cube pruning and cube growing approaches (Gesmundo and Henderson, 2010; Huang and Chiang, 2005) for enumerating k best derivations.

Lemma 3 implies that we can efficiently enumerate the top-k cost classes for a vertex x by maintaining a priority queue of the potential best derivations that allows us to walk from the optimal class $\mathcal{D}_0(x)$ with $\vec{d} = \vec{0}$ to higher cost classes. The priority queue is initially populated with $\{(e, \vec{0})\}_{e \in \mathrm{BS}(x)}$. When a derivation is removed from the queue, its neighbors are added to the priority queue sorted by their cost, and this process continues until all derivations have been exhausted or until the top-k cost classes have been enumerated. Lemma 3 guarantees that all items in cost class $j+1$ will be processed after those in cost class j. Algorithm 1 formalizes this process. The process can be made even more efficient using the faster cube pruning approach introduced by Gesmundo and Henderson (2010).

To compute $|\mathcal{D}_j(x)|$ for all hypervertices x, we process hypervertices in topological order starting from the leaves and moving up the hypergraph. Each leaf has only two derivations. The cost classes for non-leaf hypervertices can be computed via Algorithm 1. As the cost function is monotonically increasing within each edge, to obtain the top- k cost classes at a vertex x, it will always be sufficient to have computed the top- k cost classes for all of x's preceding vertices. This algorithm is similar to Algorithm 2 from Huang and Chiang (2005), except that cost classes of derivations with equivalent costs using different hyperarcs are merged. A simple example of this algorithm is illustrated in Supplementary Figure S4

Algorithm 1: Top-k Algorithm

```
input  : $x, k$
output: Vector $\mathbf{C}$ of the top $k$ cost classes of $x$
$Q = \emptyset$;
foreach $e \in BS(x)$ do
    $Q$.insert($(e, 0)$);
$\mathbf{C} = \emptyset$;
while $|Q| > 0$ and $|\mathbf{C}| < k$ do
    $(e, i) = Q$.pop();
    $L = |\mathbf{C}| - 1$;
    // If this derivation has the same cost as the last
    if $c((e, i)) = c(\mathbf{C}[L])$ then
        merge($\mathbf{C}[L], (e, i)$) ;                 // Then merge their counts
    else
        append($\mathbf{C}$, CostClass($(e, i)$));
    foreach $(e, i') \in \mathcal{N}((e, i)) \setminus Q$ do
        $Q$.insert($(e, i')$);
return $\mathbf{C}$;
```

3.5 Estimating probabilities of network history events

We now describe an algorithm that can use the counts derived via the algorithm in Section 3.4 to estimate probabilities for network history events (i.e. the interaction state or relative duplication order of ancestral proteins) based on how often they occur in the ensemble of near-optimal solutions. At a high level, the algorithm distributes a probability mass at hypervertices in accordance with how frequently they appear in near-optimal solutions.

Assigning weights to cost classes. Events that occur in derivations in low-cost classes are intuitively more believable than those that occur in very high-cost classes. We must decide the relative weight placed on these classes. If there is only a single cost class, all of the weight is assigned to the solutions from the class. Otherwise, a cost class $\mathcal{D}_j(x)$ of cost $C_j(x)$ is assigned weight using following equation involving a user-provided parameter γ:

$$w(j, x) = \frac{1}{\mathcal{Z}_x} \exp\left(\gamma \frac{x_{\min} - C_j(x)}{x_{\max} - x_{\min}}\right) \tag{11}$$

where $\mathcal{Z}_x = \sum_s \exp\left(\gamma \frac{x_{\min} - s}{x_{\max} - x_{\min}}\right)$ is a normalizing constant, s ranges over the costs of all cost classes associated with vertex x, and $x_{\min}$ and $x_{\max}$ are shorthand for the minimum and maximum costs for the computed cost classes of x. When γ is large, cost classes are given near-equal weight. At low γ, high-cost classes count for little.

Assigning probabilities to hyperarcs. Algorithm 2 traverses H in reverse topological order starting from the root. For every hyperarc $e = x \leftarrow \vec{t}$, it computes a probability $p_{\text{arc}}[e]$ that is equal to the sum of the fractions of time that this arc was used in each cost class, with each cost class weighted according to function w aforementioned. Specifically, let

$$p^j_{\text{arc}}[x \leftarrow \vec{t}] = \left(\frac{\#(e, j)}{|\mathcal{D}_j(x)|}\right) \tag{12}$$

be the conditional probability that a derivation of hypervertex x will use hyperarc $e = x \leftarrow \vec{t}$ given that the derivation is of cost $C_j(x)$. $\#(e, j)$ gives the number of times e was used in a derivation in $\mathcal{D}_j(x)$. Then, the total probability of hyperarc $x \leftarrow \vec{t}$ is given as

$$p_{\text{arc}}[x \leftarrow \vec{t}] = \sum_j w(j, x) \times p^j_{\text{arc}}[x \leftarrow \vec{t}], \tag{13}$$

where the sum runs over cost classes at x. Therefore, the probability mass contributed to hyperarc e by cost class $\mathcal{D}_j(x)$ is the weight of this cost class times the conditional probability that e was used in a derivation in $\mathcal{D}_j(x)$.

Assigning probabilities to hypervertices. The probability assigned to a hypervertex x is the sum, over all hyperarcs e where $x \in \mathbf{t}(e)$, of the probability of the hypervertex $\mathbf{h}(e)$ times $p_{\text{arc}}[e]$. That is, for every hyperarc e with x appearing in its tail, the probability mass deposited at x by e is the total probability of the head (say, y) of this hyperarc times the probability that e is used in a near-optimal derivation of y. In actuality, two probabilities, an 'in' and 'out' probability are computed for each vertex. This is described in greater detail in Supplementary Section S2.

Algorithm 2: Probability Estimation Algorithm

```
input  : Hypergraph $H$, with top-$k$ cost classes.
output: Relative frequencies for all ancestral states.
$p_{\text{in}} = \mathbf{0}$; $p_{\text{out}} = \mathbf{0}$; maxClass $= \mathbf{0}$; maxClass$[r] = k$;
$p_{in}[r] = 1.0$;
foreach $x \in V_H$ in reverse topological order do
    $p_{\text{arc}} = \mathbf{0}$;
    foreach $0 \le i <$ maxClass$[x]$ do
        foreach $e \in \mathcal{D}_i(x)$ do
            $p_{\text{arc}}[e] = p_{\text{arc}}[e] + w(i, x) \times p^i_{\text{arc}}[e]$;
            $f_e = $ frontier($\mathcal{D}_i(x), e$);
            foreach $y \in \boldsymbol{t}(e)$ do
                maxClass$[y] = \max($maxClass$[y], f_e(y) + 1)$;
    foreach $0 \le i <$ maxClass$[x]$ do
        foreach $e \in \mathcal{D}_i(x)$ do
            $z = \mathbf{res}(x, e)$;
            $a = w(i, x) \times p^i_{\text{arc}}[e]$;
            $p_{\text{out}}[z] = p_{\text{out}}[z] + p_{\text{in}}[x] \times a$;
            foreach $y \in e$ do
                $p_{\text{in}}[y] = p_{\text{in}}[y] + p_{\text{in}}[x] \times a$;
```

3.6 Predicting interactions

As the hypergraph encodes as its vertices all potential protein interactions—both extant and ancestral—the task of predicting scores for such interactions is straightforward. After running Algorithm 2, to determine a probability for edge $\{u, v\}$ existing, we simply look at the 'out' probability assigned to hypervertex $x = (\{u, v\}, \textbf{present})$. Note that the pair $\{u, v\}$ may not be considered in all potential histories, as different relative duplication orders may lead to histories in which u and v never co-exist. However, if one assumes that u and v co-exist, one can condition the relevant probabilities based on that assumption and compute the conditioned probability $p'_{\text{out}}[x] = p_{\text{out}}[x]/(p_{\text{out}}[x] + p_{\text{out}}[\bar{x}])$, where $\bar{x} = (\{u, v\}, \textbf{absent})$. Interestingly, one can use these same probabilities to compute a probability, according to the ensemble of parsimonious histories, which a pair of proteins actually co-existed.

Probabilities are also computed for extant pairs of proteins. One way to view these scores is as a phylogenetic smoothing of the input networks. This suggests that we may use the output scores of potential interactions to identify specific interactions that we would or would not expect to see given the duplication histories and the rest of the observed interactions.

To predict potential extant interactions, we consider pairs of proteins with no interaction in the input data but between which the probability output by Algorithm 2 is relatively high. For example, if interlogs exist for the potential edge in evolutionarily close species, then we expect that the derivations for a reasonable fraction of parsimonious and near-parsimonious histories will rely on the **present** interaction state between these two proteins (even though, taken in isolation, the **present** state will have a higher cost than the **absent** state). Thus, the algorithm is using all of the information encoded by the input interactions and the protein phylogeny to jointly determine the probability with which we expect to observe a given edge.

3.7 Estimating relative duplication order

We can also use the ensemble of parsimonious histories encoded by our method to compute a probability for the relative duplication order of a pair of ancestral proteins u and v. Let

$$P_u = \sum_{\substack{s\in\{\textbf{present},\,\textbf{absent}\}\\ w\in(\mathrm{anc}(v)\cup\{v\})\setminus\mathrm{anc}(u)\\ \text{with } \mathrm{sp}(w)=\mathrm{sp}(v)}} \left(p'_{\mathrm{out}}[(\{u_L, w\}, s)] + p'_{\mathrm{out}}[(\{u_R, w\}, s)]\right),$$

where anc$(\cdot)$ denotes the set of ancestors of a protein. P_u is simply the sum of probabilities that the children of u existed before the children of v, v itself, or any ancestor of v. P_v is defined analogously, swapping the roles of u and v. Then P_u represents the sum of probabilities over parsimonious histories that u duplicated before v, whereas P_v represents the probability, in our ensemble, that v duplicated before u. Thus, to predict the relative duplication order of u and v, we can simply compare the probabilities P_u and P_v and predict that the protein having the larger of the two probabilities was the first to duplicate.

3.8 Data and testing methodology

3.8.1 bZIP transcription factors To evaluate the ancestral network reconstruction task, we use the bZIP family of proteins. Similar tests were first performed by Pinney *et al.* (2007), who produced these data. The bZIP transcription factors make an enticing set of data on which to test methods for ancestral network reconstruction because the interactions between these transcription factors are strongly mediated by their coiled-coil leucine zipper domains, and the strength of these interactions can be computationally predicted with high sensitivity and specificity using sequence alone (Fong *et al.*, 2004). This means that the interaction affinity of ancestral proteins can be estimated with reasonably high confidence by first estimating the ancestral sequence and then performing a sequence-based prediction of the interaction affinity between the ancestral protein sequences. This sequence-based method was used to predict the interaction strength between both extant and inferred ancestral bZIP proteins sequences. The predicted affinities among present-day proteins were used to generate the extant interactions. Affinities among ancestral proteins were taken as the 'ground truth' ancestral interactions (Pinney *et al.*, 2007).

We experiment with three different variations on these data. The original data consist of interaction scores as predicted by the software of Fong *et al.* (2004). This software computes a score for each pair of proteins, which predicts the affinity of their potential interaction. Higher scores are assigned to pairs of proteins for which the model predicts a greater propensity for a strong interaction between these proteins. Present-day interactions were created between those pairs for which the interaction score is $\geq$30.6 (the score for which the probability of an interaction existing given the score is 0.5) (Pinney *et al.*, 2007). To create two noisy versions of the data, Gaussian noise with mean 0 and standard deviations of 10 and 20 was added to the original scores [which were in the range $(-42.87, 59.18)$], which were then converted to binary interactions as in the original dataset.

3.8.2 Herpes viruses

Protein interaction data. We use the whole proteome interaction networks of five different herpes viruses experimentally determined by Fossum *et al.* (2009). The viruses are the Epstein–Barr virus, herpes simplex virus 1 (HSV-1), murine cytomegalovirus (mCMV), Kaposi's sarcoma-associated herpesvirus (KSHV) and the varicella-zoster virus. Together, the viruses span the α, β and γ herpesvirus subfamilies and represent a sampling of viruses, which have diverged substantially, as the speciation of their common ancestor ~400 M years ago (McGeoch and Gatherer, 2005; McGeoch *et al.*, 2006). Despite this divergence, there is still a set of core orthologs that are present in all of the species.

Gene and species trees. We use the species tree representing the relationships between the five herpes virus species given by (McGeoch and Gatherer, 2005; McGeoch *et al.*, 2006). For each of the proteins in the core orthology groups assigned by Fossum *et al.* (2009), we obtained the sequences from the UniProt database (The UniProt Consortium, 2012). We then constructed gene trees for each of the orthology groups using PyCogent (Knight *et al.*, 2007). Finally, the gene trees were rooted and reconciled with the species tree using the Notung 2 software (Durand *et al.*, 2006; Vernot *et al.*, 2008).

Leave-one-out cross-validation on pairs of orthology groups. Given the reconciled gene trees for each core orthology group and the high-confidence interactions reported by Fossum *et al.* (2009), we perform our cross-validation experiments as follows. Let O denote the set of core orthology groups, and for each pair $\{a, b\}$ of groups in $O \times O$, let I_{ab} denote the set of interactions within and between groups a and b. For each pair $\{a, b\}$ of orthology groups where $|I_{ab}| > 1$, we remove each interaction i in I_{ab} in turn, while leaving the remaining interactions fixed. This yields a problem instance consisting of the reconciled trees T_a and T_b for orthology groups a and b, and the set of interactions $I_{ab} \setminus \{i\}$. We run SOPH on this instance, and we record the score assigned to each potential interaction. We rank the potential interactions according to their probabilities, and we report the relative rank of i, the left-out interaction, among the list of potential, non-input interactions. In other words, let L_a and L_b denote the leaf nodes of T_a and T_b (not considering nodes marked as lost by the reconciliation algorithm) and $L_{ab} = L_a \cup L_b$. Then, we consider all potential interactions $i' \in P_{ab}$, where $P_{ab} = \{\{u, v\} | u, v \in L_{ab} \wedge \mathrm{sp}(u) = \mathrm{sp}(v)\} \setminus (I_{ab} \setminus \{i\})$, and sort them in descending order according to their assigned scores. The requirement that $\mathrm{sp}(u) = \mathrm{sp}(v)$ enforces that $\{u, v\}$ is only a potential interaction if u and v belong to the same species. We compute the relative rank of i in this list as $\mathrm{rank}_{\mathrm{rel}}(i) = \mathrm{rank}(i)/(|P_{ab}| - 1)$. Ideally, the relative rank should be low, indicating that the left-out edge was near the top of the list of predicted interactions. The relative rank is always in the range of 0–1 (inclusive), and if the ranks were assigned randomly, we would expect the left-out interaction to have relative rank of 0.5 on average (this property holds empirically).

4 RESULTS

We now describe the performance of the SOPH framework for the three inference tasks—ancestral network reconstruction, missing interaction imputation and determination of relative protein duplication order—set forth in Section 1.

4.1 Reconstructing bZIP ancestral networks in the presence of noise

For each of the noise levels of the input data ($\sigma = 0, 10, 20$), we reconstruct three ancestral networks—Teleost (ancestor of *Danio rerio* and *Takifugu rubripes*), vertebrata (ancestor of *D.rerio*, *T.rubripes* and *Homo sapiens*) and chordate (ancestor of *D.rerio*, *T.rubripes*, *H.sapiens* and *Ciona intestinalis*). For all experiments, we use the top $k = 40$ cost classes and set γ, the parameter that determines the relative weight of the different cost classes to $1.5k = 60$.

To measure the quality of the ancestral network reconstruction, we use three separate metrics, the BEDROC score

Table 1. Ancestral network reconstruction accuracy of several methods under various levels of noise σ

Ancestor	Method	BEDROC ($\sigma = 0, 10, 20$)	AUROC	AUPR
Vertebrata	SOPH	**0.96, 0.9, 0.83**	0.96, 0.91, **0.88**	**0.75, 0.64, 0.55**
	Parsimony	0.89, 0.8, 0.7	0.84, 0.77, 0.78	0.68, 0.58, 0.52
	Probabilistic	0.83, 0.76, 0.7	0.96, 0.91, 0.86	0.63, 0.53, 0.43
Teleost	SOPH	**0.95, 0.9, 0.81**	0.97, 0.94, 0.9	**0.76, 0.68, 0.57**
	Parsimony	0.84, 0.78, 0.66	0.87, 0.8, 0.8	0.67, 0.6, 0.53
	Probabilistic	0.88, 0.82, 0.71	0.97, 0.94, 0.9	0.7, 0.6, 0.47
Chordata	SOPH	**0.97, 0.93, 0.75**	0.95, 0.88, 0.85	**0.7**, 0.6, **0.44**
	Parsimony	0.87, 0.86, 0.56	0.73, 0.75, 0.60	0.59, 0.58, 0.35
	Probabilistic	0.93, 0.92, 0.68	0.95, **0.93, 0.88**	0.67, **0.63**, 0.43

Note: The performance of our SOPH approach, a single-history parsimony approach (Patro *et al.*, 2012) and the probabilistic method described by Pinney *et al.* in reconstructing the ancestral interaction networks we consider.

(Truchon and Bayly, 2007), the area under the ROC (AUROC) and the area under the precision-recall curve (AUPR). The BEDROC metric is an AUC metric meant to deal with the so-called early enrichment or early recognition problem. Intuitively, the BEDROC metric weights the accuracy more heavily early on in the retrieval list. This is appropriate for the task of inferring ancestral interactions because we expect the density of such interactions to be relatively low, and because we care most about those inferred ancestral interactions in which we have high confidence. When computing BEDROC scores, we set the early-recall parameter α to 20.0 as suggested by Truchon and Bayly (2007).

Table 1 demonstrates the performance of our ancestral network reconstruction procedure compared with the single-history parsimony approach (Patro *et al.*, 2012) and the probabilistic model used by Pinney *et al.* (2007). We find that our method often outperforms both other methods under the three metrics shown in Table 1.

These results show the potential benefit of using the SOPH approach to the ancestral network reconstruction problem, especially in the typical situation where the error rates of measured present-day interactions can be very high (Stumpf *et al.*, 2007). More generally, the results demonstrate that the probabilistic method, although clearly more robust to noise than the naïve parsimony approach, is not inherently superior in this aspect to advanced methods based on parsimony. In particular, the results of the SOPH approach with noisy input interactions suggests that a method based on analyzing an ensemble of parsimonious solutions can exceed the accuracy of methods based on maximum likelihood. By exploring all near-optimal parsimonious histories, SOPH is able to overcome one of the main shortcomings of previous parsimony-based approaches and to provide substantially better performance, in most cases, than any of the pre-existing methods.

We note that the cases in which the maximum-likelihood approach is most competitive with SOPH is in the most ancient ancestral species. However, this is also the species in which we have the least confidence in the ground-truth data, as ground-truth ancestral interactions were computed based on the interaction scores of inferred ancestral sequences.

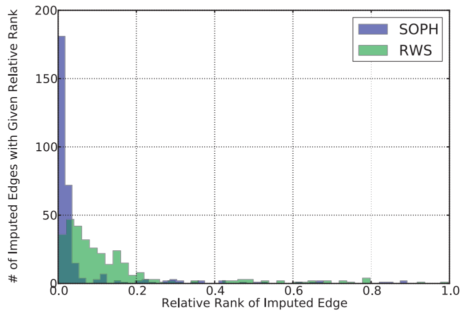

Fig. 2. A histogram of the relative ranks of the left-out edges in 10-fold cross-validation experiments on the bZIP network

We cannot perform a similarly exhaustive validation of the ancestral predictions for the herpes virus networks because we do not have a general scheme for determining ground-truth ancestral interactions. Anecdotally, however, we note that the second highest probability ancestral interaction predicted by our method in the common ancestor of all five herpes virus species was between the orthology groups containing HSV-1 proteins UL33 and UL31. The interactions between these orthology groups were posited by Fossum *et al.* (2009) to be highly conserved in their study of evolutionarily conserved protein interaction in the herpes networks; suggesting that this interaction likely did exist in the ancestral network.

4.2 Imputing missing present-day bZIP interactions

We also test the accuracy of SOPH for predicting missing extant interactions in the present-day bZIP networks. Let L denote the set of leaves (i.e. extant proteins) in T, and let I be the set of ground-truth interactions among L. We define $U = \{\{u, v\} \in L \times L | \text{species}(u) = \text{species}(v)\}$ as the universe of potential interactions.

We performed leave-one-out (LOO) and 10- and 5-fold cross-validation (CV) to test the accuracy of our imputations. In LOOCV, the mean relative rank of the left-out interaction is 0.05 and the median relative rank is 0.01. We observe a minor decrease in performance, with mean ranks of 0.06 and 0.08 and median ranks of 0.01 and 0.02, when simulating lower data

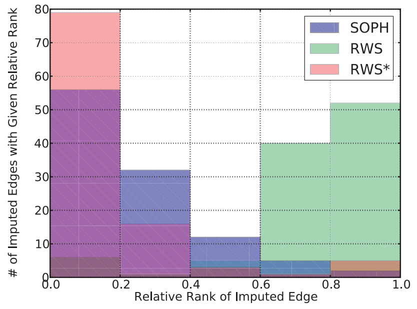

Fig. 3. Histogram of the ranks of the left-out edge in cross-validation experiments for the herpes virus networks. RWS denotes the standard RWS method, which always predicts homodimer interactions. As the core herpes network has few such interactions, the resulting predictions are poor. For RWS*, all homodimer predictions produced by RWS were set to 0. This extra information substantially improves the RWS predictions; however, such information is usually not known when performing an edge imputation task. SOPH, on the other hand, can effectively predict homodimer interactions on a protein-by-protein basis

coverage with 10- and 5-fold cross-validation. Alternatively, the edges predicted by the RWS (Lei and Ruan, 2013) method have higher mean relative ranks of 0.12, 0.14 and 0.18 and median relative ranks of 0.08, 0.08 and 0.1 on LOO, 10-fold and 5-fold cross-validation tests, respectively.

The histogram of relative ranks among all experiments (10-fold CV results; Fig. 2) displays a highly skewed distribution for both methods, but SOPH clearly assigns relative ranks closer to 0 (the optimum) for most edges. In fact, with the SOPH predictions, the vast majority of the testing edges appear in the top 2% of the potential edges, and the frequency of lower probabilities for the true left-out edge falls off exponentially. This suggests that our algorithm is able to identify missing present-day edges with high accuracy.

4.3 Imputing present-day interactions in herpes viruses

We also applied our network history inference framework to predict missing edges in herpes viruses. Unfortunately, the core orthology groups are small enough, and the interactions between and within them sparse enough, that the testing methodology precludes anything other than leave-one-out cross-validation (described in Section 3.8). We test the relative ranks of the SOPH predictions, as well as those of RWS and a variant thereof (RWS*) where predicted self-loops are removed in a post-processing step. Although our method only considers the interactions within and between each pair of orthology groups in isolation, RWS is provided with the entire core interaction network for each test. The distributions of relative ranks for the three different methods are shown in Figure 3.

Across all homology groups, the relative ranks computed by SOPH for the left-out interactions are substantially lower than we would expect by chance, with a mean relative rank of 0.23, and median rank of 0.16, indicating that the left-out edge is nearly always in the top 25% of the possible edges. The unmodified RWS predictions obtain mean relative rank of 0.78 and median of 0.79. This is primarily because of the fact that RWS always predicts the existence of a homodimer interaction. In

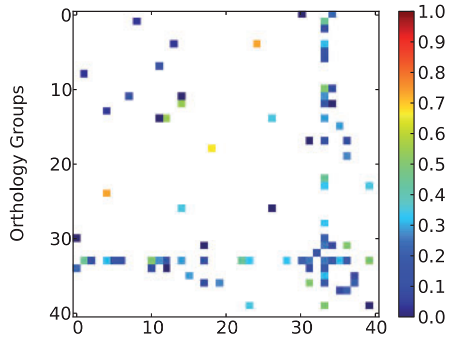

Fig. 4. Imputing missing interactions (per-group). A heatmap of the average relative ranks of the left-out edge between pairs of orthology groups

certain protein families with a high homodimerization rate (like bZIP), such predictions are often accurate. However, in the herpes networks, where homodimers are rare, these predictions are problematic for RWS. Thus, we also tested a variant of the RWS predictions (RWS*) where all predicted homodimer scores were set to 0 in a post-processing step. This results in a mean relative rank of 0.14 and a median relative rank of 0; a huge improvement in performance over the unmodified RWS predictions. Setting any homodimer scores to 0 also improves the SOPH predictions, but not as drastically. However, such information about the homodimerization rate is not often known a priori, and one strategy is not always better than the other (e.g. RWS outperforms RWS* in bZIP). Thus, although RWS either predicts the existence of all or no homodimer interactions, SOPH can predict them effectively on a protein-by-protein basis.

Some of the cases in which the experimentally deleted edge is given a low probability (by either method) are likely because of interactions, which are surprising from an evolutionary perspective, or simply a result of the sparsity of the input dataset. In particular, as the experimental dataset used to perform these tests is hypothesized to have a relatively high–false-negative rate itself (Fossum *et al.*, 2009), it is likely the case that the evolutionary evidence to improve the prediction of edges is simply missing.

Performance on individual pairs of orthology groups. Figure 4 provides a heatmap of the average relative rank of imputed interactions between pairs of orthology groups. Because of the sparsity of the initial data and the presumed low density of the true interaction networks, many pairs of orthology groups contain one or zero interactions between them (they appear white in Fig. 4) and are left-out of the experiment. Among the remaining groups, we notice a somewhat bimodal distribution of relative ranks. Between many pairs of groups, the missing interactions can be perfectly imputed (relative rank of 0), whereas between others, the task seems incredibly difficult (e.g. between groups 4 and 24 the average relative rank of the left-out edge was 0.63). Again, this suggests that when there is sufficient evolutionary evidence, missing interactions can be imputed with high accuracy. Because we do not have a true gold-standard set of interactions, we cannot reliably hypothesize whether the imputed interactions with large relative ranks are because of a failure of the method (i.e. evolutionarily non-parsimonious interactions) or simply false-negatives in the input data.

4.4 Inferring relative order of duplications

Because probabilities for several kinds of evolutionary events can be extracted from the ensembles of histories, the method can also be used to estimate the relative duplication order of ancestral proteins as described in Section 3.7. Accurate independent measurements of duplication order can help validate branch lengths and also estimate the relative ordering of speciation events.

We will compare our inferred duplication order with the duplication order implied by an ultrametric embedding of the bZIP phylogeny. The branch lengths inferred on the original bZIP tree [constructed via PAML (Yang, 1997)] do not satisfy the ultrametric property; thus, they are not consistent with a molecular clock and cannot be directly used to infer duplication order. By embedding the given branch lengths into an ultrametric tree using the method provided in Huerta-Cepas *et al.* (2010), we obtain consistent relative orderings based on sequence information alone.

To measure the agreement between the orders inferred by SOPH and those inferred by sequence, we compute the standard Kendall $\tau - b$ statistic among all intra-species pairs of gene duplication events that were not related by direct evolution. Let N be the total number of tested pairs, n_c be the number of such pairs that the two methods place in the same relative ordering, n_d be the number of pairs for which they disagree, t_x be the number of tied pairs given the tree ordering and t_y be the number of tied pairs given the SOPH ordering. Then $\tau - b = (n_c - d_d)/\sqrt{(N - t_x)(N - t_y)}$.

Among all 5194 relevant pairs, there are 3745 concordant and 1349 discordant pairs, leading to $\tau - b = 0.47$. This correlation is highly significant, with a P-value of 2×10^{-12}, using the analytic estimation of variance suggested in Hazewinkel (2000). When performing the aforementioned test, we did not supply SOPH with the branch lengths, and the method did not use any information about the relative duplication order or protein sequences apart from the ancestral relationships encoded in the tree topologies themselves. This relatively good performance means that there is a substantial amount of information about the relative duplication order of proteins encoded in the network. The relatively high $\tau - b$ and low P-value indicate that the SOPH approach is able to reconstruct, in a largely independent way, the relative order of duplication events.

We note that if we use the duplication order implied by the non-ultrametric version of the bZIP phylogeny—where the branch lengths still encode evolutionary information but cannot be interpreted directly as representing evolutionary time—we obtain a $\tau - b$ of 0.20 (3109 concordant and 2084 discordant pairs) and an associated P-value of 0.002. This suggests that SOPH can be useful in providing a separate and not-often considered source of information (extant interaction networks) when attempting to determine a consistent set of branch lengths and duplication orders.

5 CONCLUSION

We have introduced a novel sum-over-histories method for solving the network history inference problem. It addresses shortcomings of existing methods by using a weighted ensemble consisting of all optimal and near-optimal parsimonious histories. We show that this makes the results robust to the presence of noise in the input (Section 4.1) and allows our parsimony approach to outperform the probabilistic approach to ancestral network reconstruction (Pinney *et al.*, 2007) at all considered noise levels.

The algorithms we present have practical running times. Our implementation required only 1.5 min to compute—in serial—the results for all cross-validation experiments on the herpes virus datasets (an average of <1 s per experiment). On the significantly larger bZIP dataset, the algorithm requires 6.5, 8.5, 10.4, 12.4, 14.4, 16.9 and 34.4 s, respectively, to compute solutions using the top 1, 10, 20, 30, 40, 50 and 100 cost classes, suggesting an empirically linear relationship between the running time and the number of requested cost classes.

The sum-over-histories approach is also general and allows many other questions to be answered about how a sequence of proteins, and their interactions have evolved. We find that our method can reliably exploit evolutionary evidence to discover the existence of missing interactions. SOPH may be useful in prioritizing low-throughput but high-accuracy protein interaction experiments by suggesting which interactions are more likely than others to exist given the current experimental and evolutionary evidence. The SOPH approach also recovers the ultrametric temporal ordering relationships between duplication events well, without using any direct information about sequence or branch lengths.

ACKNOWLEDGEMENT

The authors thank Dan Brown and Yann Ponty for helpful discussions and John Pinney for providing the bZIP data.

Funding: National Science Foundation (CCF-1256087, CCF-1053918, EF-0849899 and IIS-0812111); National Institutes of Health (1R21AI085376); University of Maryland Institute for Advanced Studies New Frontiers Award. C.K. received support as an Alfred P. Sloan Research Fellow.

Conflict of Interest: none declared.

REFERENCES

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

Borenstein,E. *et al.* (2008) Large-scale reconstruction and phylogenetic analysis of metabolic environments. *Proc. Natl Acad. Sci. USA*, **105**, 14482–14487.

Carvalho,L. and Lawrence,C. (2008) Centroid estimation in discrete high-dimensional spaces with applications in biology. *Proc. Natl Acad. Sci. USA*, **105**, 3209–3214.

Durand,D. *et al.* (2006) A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J. Comp. Biol.*, **13**, 320–335.

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

Feynman,R. (1942) *The Principle of Least Action in Quantum Mechanics*. PhD Thesis, Princeton University, USA [Feynman's Thesis: a New Approach to Quantum Theory, World Scientific 2005].

Finkelstein,A. and Roytberg,M. (1993) Computation of biopolymers: a general approach to different problems. *Biosystems*, **30**, 1–19.

Flannick,J. *et al.* (2006) Graemlin: general and robust alignment of multiple large interaction networks. *Genome Res.*, **16**, 1169–1181.

Flannick,J. *et al.* (2009) Automatic parameter learning for multiple local network alignment. *J. Comput. Biol.*, **16**, 1001–1022.

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.

Gallo,G. *et al*. (1993) Directed hypergraphs and applications. *Discrete Appl. Math*., **42**, 177–201.

Gesmundo,A. and Henderson,J. (2010) Faster cube pruning. In: Federico,M. *et al*. (eds) *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*. IWSLT, Paris, France, pp. 267–274.

Gibson,T.A. and Goldberg,D.S. (2009) Reverse engineering the evolution of protein interaction networks. *Pac. Symp. Biocomput*., 190–202.

Hazewinkel,M. (2000) Encyclopaedia of mathematics: an updated and annotated translation of the Soviet 'Mathematical encyclopaedia'. In: *Encyclopaedia of Mathematics*. Vol. 12, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Huang,L. and Chiang,D. (2005) Better k-best parsing. In: *Proceedings of the Ninth International Workshop on Parsing Technology*. Association for Computational Linguistics Stroudsburg, PA, USA, pp. 53–64.

Huerta-Cepas,J. *et al*. (2010) ETE: a python environment for tree exploration. *BMC Bioinformatics*, **11**, 24.

Klein,D. and Manning,C.D. (2001) Parsing and hypergraphs. In: *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001)*, Tsinghua University Press, Beijing, China.

Knight,R. *et al*. (2007) PyCogent: a toolkit for making sense from sequence. *Genome Biol*., **8**, R171.

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

Lei,C. and Ruan,J. (2013) A novel link prediction algorithm for reconstructing protein–protein interaction networks by topological similarity. *Bioinformatics*, **29**, 355–364.

Li,S. *et al*. (2012) Reconstruction of network evolutionary history from extant network topology and duplication history. In: *Proceedings of the 8th International Conference on Bioinformatics Research and Applications*. Springer-Verlag, Berlin, Heidelberg, pp. 165–176.

McGeoch,D.J. and Gatherer,D. (2005) Integrating reptilian herpesviruses into the family herpesviridae. *J. Virol*., **79**, 725–731.

McGeoch,D.J. *et al*. (2006) Topics in herpesvirus genomics and evolution. *Virus Res*., **117**, 90–104.

Middendorf,M. *et al*. (2005) Inferring network mechanisms: the *Drosophila melanogaster* protein interaction network. *Proc. Natl Acad. Sci. USA*, **102**, 3192–3197.

Mithani,A. *et al*. (2009) A stochastic model for the evolution of metabolic networks with neighbor dependence. *Bioinformatics*, **25**, 1528–1535.

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

Nielsen,L.R. *et al*. (2005) Finding the k shortest hyperpaths. *Comput. Oper. Res*., **32**, 1477–1497.

Patro,R. *et al*. (2012) Parsimonious reconstruction of network evolution. *Alg. Mol. Biol*., **7**, 25.

Pereira-Leal,J.B. *et al*. (2007) Evolution of protein complexes by duplication of homomeric interactions. *Genome Biol*., **8**, R51.

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.

Ponty,Y. and Saule,C. (2011) A combinatorial framework for designing (pseudoknotted) RNA algorithms. In: *WABI*. Springer, Berlin Heidelberg, pp. 250–269.

Singh,R. *et al*. (2007) Pairwise global alignment of protein interaction networks by matching neighborhood topology. In: *Proceeding. International Conference on Research in Computational Molecular Biology (RECOMB)*. Springer Berlin Heidelberg, pp. 16–31.

Stumpf,M.P. *et al*. (2007) Evolution at the system level: the natural history of protein interaction networks. *Trends Ecol. Evol*., **22**, 366–373.

The UniProt Consortium. (2012) Reorganizing the protein space at the universal protein resource (UniProt). *Nucleic Acids Res*., **40**, D71–D75.

Truchon,J.F. and Bayly,C.I. (2007) Evaluating virtual screening methods: good and bad metrics for the 'early recognition' problem. *J. Chem. Inf. Model*., **47**, 488–508.

Vernot,B. *et al*. (2008) Reconciliation with non-binary species trees. *J. Comput. Biol*., **15**, 981–1006.

Yang,Z. (1997) Paml: a program package for phylogenetic analysis by maximum likelihood. *Comput. Appl. Biosci*., **13**, 555–556.

Zhang,X. and Moret,B.M. (2008) Boosting the performance of inference algorithms for transcriptional regulatory networks using a phylogenetic approach. In: *Proceedings International Workshop on Algorithms in Bioinformatics (WABI)*, BioMed Central, London, pp. 245–258.

Zhang,X. and Moret,B. (2010) Refining transcriptional regulatory networks using network evolutionary models and gene histories. *Alg. Mol. Biol*., **5**, 1.

Zhu,Y. and Nakhleh,L. (2012) Reconstructing the evolution of molecular interaction networks under the DMC and link dynamics models. In: *Algorithms in Bioinformatics*. Springer, Berlin Heidelberg, pp. 57–68.