BIOINFORMATICS ORIGINAL PAPER Vol. 28 no. 9 2012, pages 1202–1208
doi:10.1093/bioinformatics/bts109

Phylogenetics Advance Access publication March 6, 2012

Mirroring co-evolving trees in the light of their topologies

Iman Hajirasouliha[1,†], Alexander Schönhuth[2,†], David de Juan[3], Alfonso Valencia[3] and S. Cenk Sahinalp[1,*]

[1]School of Computing Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada, [2]Centrum Wiskunde & Informatica (CWI), Science Park 123 1098 XG, Amsterdam, The Netherlands and [3]Structural Biology and Biocomputing Programme (CNIO) Spanish National Cancer Research Centre, 28029 Madrid, Spain

Associate Editor: David Posada

ABSTRACT

Motivation: Determining the interaction partners among protein/domain families poses hard computational problems, in particular in the presence of paralogous proteins. Available approaches aim to identify interaction partners among protein/domain families through maximizing the similarity between trimmed versions of their phylogenetic trees. Since maximization of any natural similarity score is computationally difficult, many approaches employ heuristics to evaluate the distance matrices corresponding to the tree topologies in question. In this article, we devise an efficient deterministic algorithm which directly maximizes the similarity between two leaf labeled trees with edge lengths, obtaining a score-optimal alignment of the two trees in question.

Results: Our algorithm is significantly faster than those methods based on distance matrix comparison: 1 min on a single processor versus 730 h on a supercomputer. Furthermore, we outperform the current state-of-the-art exhaustive search approach in terms of precision, while incurring acceptable losses in recall.

Availability: A C implementation of the method demonstrated in this article is available at http://compbio.cs.sfu.ca/mirrort.htm

Contact: imanh@sfu.ca; cenk@sfu.ca; as@cwi.nl

Received on October 18, 2011; revised on February 23, 2012; accepted on February 24, 2012

1 INTRODUCTION

The vast majority of cellular functions are exerted by combinations of interacting gene products. As a result, 'preservation of functionality' among proteins and other gene products typically implies 'preservation of interactions' across species. It is well established that protein–protein interactions (both physical interactions as well as co-occurence of domains) are preserved through speciation events [see Lovell and Robertson (2010); Pazos and Valencia (2008) and the references therein]. A major implication of this is that the evolutionary trees behind two interacting protein families can look near-identical.

As interacting proteins have a tendency to co-evolve, it may be possible to assess the potential of two or more proteins (or other gene products) being interaction partners by measuring how similarly they evolve across related species. For this purpose, a number of computational strategies have been developed. Such strategies aim to compare the phylogenetic trees of two (or more) protein or protein-domain families, where paralogs and orthologs are represented with leaves with appropriate labels and internal vertices can be interpreted as either speciation or duplication events. Among these strategies we will focus on *mirrortree* approaches, which explicitly or implicitly map leaves of a pair of trees (belonging to two distinct proteins or gene products) onto one another such that the leaves that are mapped to each other would be identified as potential interaction partners. Mirrortree approaches aim at an overall quantification of 'family similarity' via a measure of tree similarity. Typically, these approaches do not aim to modify the specific topology of the underlying phylogenetic trees and thus are different from tree reconciliation approaches (Page, 1994; Vyugin *et al.*, 2002). They are also distinct from phylogenetic profiling methods (Pellegrini *et al.*, 1999), which aim to measure the phylogenetic profiles of proteins or domains to check for potential interaction partners.

The first mirrortree approach was proposed to discover protein–protein (rather than domain–domain) interactions and was based on comparing the distance matrices[1] resulting from the multiple alignment members of each protein family (Pazos and Valencia, 2001). Note that one can interpret this as mapping leaves onto one another, as will be explained below. Since this study, a number of mirrortree approaches have been developed; almost all of these approaches are again based on comparing distance matrices rather than the trees directly (see the introductory paper by Pazos and Valencia (2001) and Pazos and Valencia (2008) for more references). In fact, direct comparison of gene trees has been considered as '... a problem yet to be fully resolved.' (Izarzugaza *et al.*, 2008, p. 2).

In this article, we consider a fresh approach to the problem of predicting protein or other gene product interactions by comparing gene trees directly, without the aid of a distance matrix. Note that such a distance matrix is a byproduct of the underlying phylogenetic tree: popular multiple sequence alignment methods typically align sequences in the order imposed by their phylogenetic tree and the 'distances' in the matrix correspond to the distances in the phylogenetic tree. As a result our method should be considered as a more direct approach to mirroring trees.

In the case where there are no paralogs of any gene, assessing tree similarity is both computationally straightforward and reliable (Pazos and Valencia, 2008). More specifically, if there is at most one family member per species, the mapping problem reduces to

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

[1]The distance matrix of a gene tree is comprised of entries (i,j) which represent the distance between leaves i and j.

the problem of finding out species where the interaction is lost; after removal of such species, the topologies of the two trees will be identical, i.e. the leaf representing a particular species in one tree will correspond to the leaf representing the same species in the other tree.

In the presence of paralogous genes (and thus proteins or gene products), however, the mapping problem becomes much more complex. For example, if we have n paralogs per tree for one of the species, we may need to evaluate more than $n!$ many potential mappings ($n!$ is the number of mappings where each paralog from one tree pairs with a paralog from the other tree. In addition, there are mappings where one has to remove non-interacting paralogs. As was pointed out in Tillier *et al.* (2006), protein interaction can be preserved during duplication, while interaction can be lost during speciation.). Thus, the number of potential mappings is super-exponential in the number of paralogs per species, implying a significant computational challenge.

There are a number of mirrortree approaches which address the presence of paralogs and aim directly at inferring the correct mapping of leaves; these approaches typically aim to 'align' the distance matrices by shuffling and eliminating the rows (and corresponding columns) so as to maximize the similarity between the matrices. The similarity between two aligned matrices is defined in the form of root mean square difference (Ramani and Marcotte, 2003), correlation coefficient (Gertz *et al.*, 2003), information-theoretic 'total interdependency' of multiple alignments (Tillier *et al.*, 2006), Student's t (Izarzugaza *et al.*, 2008) or the size of the largest common submatrix (Tillier and Charlebois, 2009). Because an exact solution to the matrix alignment problem, where the goal is to maximize any of these notions of similarity (by determining the right mapping of rows and columns), is hard to compute, many available approaches employ heuristics based on swapping pairs of rows/columns in a greedy fashion. These methods also commonly perform column/row elimination from the 'larger' matrix only, and not the other (Gertz *et al.*, 2003; Izarzugaza *et al.*, 2008; Jothi *et al.*, 2005; Ramani and Marcotte, 2003; Tillier *et al.*, 2006). We are aware of one exception by (Tillier and Charlebois, 2009), which aims to determine the largest common (i.e. within a threshold) submatrix and removes the remainder of the columns and rows from both matrices. Similarly the only approach which directly compares the tree topologies themselves is by Jothi *et al.* (2005), which uses a Metropolis algorithm to heuristically travel 'tree automorphism' space. However, this approach cannot handle trees of different sizes. See Lovell and Robertson (2010); Pazos and Valencia (2008); and Tillier and Charlebois (2009) for references on mirrortree approaches which do not necessarily relate to the mapping problem in the presence of paralogs.

In this article, we present polynomial-time algorithms that determine mappings of leaves which respect the topologies of the two trees compared. As input, we are given two 'gene trees' T and T' of two protein/domain families known to interact with one another. T and T' have labeled leaves where labels reflect species such that the presence of the same label at two different leaves reflects the presence of paralogs. We introduce and formally define the *gene tree alignment problem*, which aims to delete both leaves and inner vertices from both trees until the remaining trees are isomorphic, that is, one can map the vertices of the two remaining trees in a one-to-one fashion onto another such that ancestor relationships are preserved. This in particular implies a one-to-one mapping of the remaining leaves, which we present as output. Clearly, there are many different possible choices of such one-to-one mappings of leaves—our algorithms determine the *score-optimal* such alignment where different deletion operations are penalized in different ways, depending on how they transform the topologies of the trees. Note that our algorithm depends on some (user-defined) cost parameters, that can be used to impose constraints on the alignment. We describe the nature of our scoring scheme in detail in the following; please see Section 3 for full details and precise notations.

Note that the algorithm only outputs one uniquely determined, score-optimal alignment of subsets of leaves of T, T'. Note further that we do not perform an exhaustive search since we never consider mappings of leaves which imply mappings of internal vertices that do not preserve ancestor relationships of the gene trees T, T' and thereby contradicts their topologies.

Our method can be viewed as an extension of tree-edit distance approaches. Alternative constraints leading to polynomial time solvable variants on the tree edit distance is surveyed in Zhang (1996). For further, more recent work see also Pinter *et al.* (2008) that address the subtree homeomorphism problem, which, given a 'text' tree T and a 'pattern tree' P as the input, asks to find a subtree t in T such that P is homeomorphic to t. Now, two trees T_1, T_2 are said to be homemorphic if one can remove degree 2 vertices from T_1, T_2 such that T_1 and T_2 are isomorphic. Another recent work (Raynal *et al.*, 2010) considers homeomorphic alignment of 'weighted' but unlabeled trees. Here the goal is to obtain a homeomorphic mapping between vertices of two trees such that the differences between the weights of 'aligned' edges is minimized. While being related to our approach, the method described in Raynal *et al.* (2010) is not applicable to our problem as the trees they consider are not leaf labeled. We refer the reader to Bille (2003) for a general and gentle overview of further related work on tree edit distance, tree alignment and tree inclusion.

The main technical contribution of this article is a novel deterministic mirrortree algorithm that directly compares tree topologies. The algorithm is optimal within the constraints we impose and is provably efficient. We compare our algorithm with the most recent, state-of-the-art heuristic search approach (Izarzugaza *et al.*, 2008) that aims to maximize the similarity between distance matrices, where distances reflect lengths of shortest paths in neighbor-joining trees. In our comparisons, we use precisely the same trees to be able to juxtapose a distance matrix-based heuristic search method to our topology-based, deterministic method without introducing further biases.

2 PRELIMINARIES AND NOTATIONS

Let $T=(V,E,w)$ be a tree with weighted edges as given by a non-negative weight function $w: E \to \mathbb{R}_+$. We denote the leaves of T by $L=\{\ell_1,\ldots,\ell_n\}$, the internal nodes of T (excluding the root) by $U=\{u_1,\ldots,u_m\}$ and the root of T by r. In particular, let n be the number of leaves and m be the number of internal vertices without the root. Note that a tree T is binary and rooted if and only if $\deg(r)=2$ and $\deg(u)=3$ for all internal vertices $u \in U$; this will imply that $m=n-2$ and $|E|=2n-2$. In our setting, edge weights $w(v_i,v_j)$ reflect the evolutionary distance between adjacent vertices v_i, v_j. Note that leaves refer to gene products whereas internal vertices can be interpreted as speciation and/or duplication events. For a given vertex $v \in V$, we define $\theta(v)$ as the evolutionary distance

between the root and v. In other words, $\theta(v)$ is the sum of the edge weights in the unique path from the root to v. In rooted trees, there is a natural partial order

$$v_i \leq v_j \quad \Leftrightarrow \quad v_i \text{ is an ancestor of } v_j \tag{1}$$

on the vertices of T. Hence, the edges have a natural orientation and each vertex v_i induces a unique subtree $T(v_i)$. This partial order is crucial for our algorithm—which cannot be applied to unrooted trees in a straightforward manner. For processing unrooted (e.g. neighbor-joining) trees, consider the pair of proteins/domains (one from each tree) which are known to interact. We root the two trees at these vertices in order to apply our algorithm. Provided such a pair exists (which is typically the case), our algorithm optimally aligns the trees as it does not assume any order among the many sibling vertices. In a tree T which is rooted at r, we call vertex u the parent of a vertex v if u and v are connected by an edge and u is closer to r than v. The *height* of a rooted tree is defined as $\max\{d(r,\ell_i)\,|\,i=1,\ldots,n\}$ where $d(v_1,v_2)$ is the length of the shortest path between vertices v_1 and v_2 without considering edge weights, that is the maximum (unweighted) distance of the root to a leaf. We denote a bijection (i.e. a one-to-one and onto alignment) of subsets of vertices of T, T' by $\mathcal{M}[T,T']$ and write

$$M := \{(v,w) \in T \times T' \,|\, \mathcal{M}(v) = w\} \tag{2}$$

for the pairs of mapped vertices. Note that in such a bijection, not all vertices of T are necessarily mapped to a vertex in T' and vice versa. We refer to vertices which are not mapped as *deleted* by $\mathcal{M}[T,T']$. We only consider alignments which satisfy the following: (i) the alignment preserves the ancestor relationship of T and T'; (ii) only leaves with identical labels are mapped onto one another; and (iii) upon deletion of vertices, where deletion of an internal vertex v leads to new edges joining the parent of v with the children of v, the two tree topologies are isomorphic. Among the alignments satisfying the above conditions, we compute the alignment that has maximum score.

For a formal definition of our scoring scheme, consider the internal vertices of T and T' that are deleted. Among them, we distinguish between vertices v that have descendants x which are not deleted. We write N_I for such vertices. We write N_T for the remaining deleted vertices. Note that each vertex $v \in N_T$ makes part of a subtree of T which has been deleted as a whole. The score of the alignment is then defined as

$$S(\mathcal{M}[T,T']) = \sum_{(v,v')\in M} S_M(v,v') + \sum_{v\in N_I} S_{N_I}(v) + \sum_{v\in N_T} S_{N_T}(v). \tag{3}$$

The individual score functions S_M, S_{N_I} and S_{N_T} will be formally defined in Section 3. As noted above, our algorithm, which maximizes the overall score of the alignment, can be viewed as an extension of the standard *tree edit distance* algorithm for unweighted trees (Tai, 1979), to those with edge weights. Determining the tree edit distance is NP-complete (Zhang *et al.*, 1992) [in fact MAX-SNP-hard (Zhang and Jiang, 1994)]. Since the instances treated here are too large (trees have up to >200 leaves) we have to impose reasonable constraints when aiming at fast, polynomial-runtime solutions. Motivated by test runs (see numbers referring to $C_{1,2}$ in Sections 4 and 5), we chose to impose the additional constraint

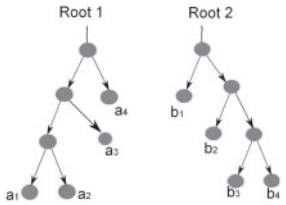

Fig. 1. Two isomorphic trees are shown as an example in this figure. The leaves of the left tree are labeled with a_1, a_2, a_3 and a_4 whereas the leaves of the tree on the right are labeled with b_1, b_2, b_3 and b_4. A possible mapping between the leaves that respect the tree topology is $(a_1,b_3),(a_2,b_4),(a_3,b_2)$ and (a_4,b_1).

that a vertex u and its parent v cannot be deleted at the same time without deleting the entire subtree rooted at v. That is we disallow to have both a parent v and a child u in N_I. Note, however, that deletion of two internal siblings is permissible—we found that such deletions can lead to favorable alignments. As the operation of deleting entire subtrees does not lead to runtime issues, does not perturb the topology of the remaining trees and also reflects the biologically reasonable assumption that interaction can be lost for entire subtrees, we allow it without additional restrictions.

3 METHODS

Given two rooted weighted-edge trees T and T', our algorithm *aligns* the trees by mapping a subset of leaves of T to a subset of leaves of T'. In order to obtain this mapping, a series of (i) individual vertex deletions or (ii) subtree deletions (with specific penalties) are performed on each tree with the goal of obtaining two isomorphic trees $T_1 = (V_1, E_1, w_1)$ (from T) and $T_1' = (V_1', E_1', w_1')$ (from T_1'); Figure 1 shows two such rooted trees that are isomorphic; it also shows a mapping between the leaves. The specifics of vertex and subtree deletions on a tree $T = (V, E, w)$ are as follows.

(1) Deleting an internal vertex v also deletes the edge (u,v), where u is the parent of v. Furthermore, it connects each child x of v to u by deleting the edge (v,x) and creating a new edge (u,x). The weight of this new edge, $w(u,x)$ is set to $w(u,v)+w(v,x)$. As mentioned earlier, it is not possible to delete both a node v and its parent u from T.

(2) Deleting an entire subtree rooted at an internal vertex v deletes all descendants of v and their associated edges.

In the remainder of this section, we will discuss the costs of the above deletion operations and the scores of the mapped vertices. As mentioned earlier, the overall score of the mapping will be the sum of the scores of the mapped vertices and the scores (negative costs) of the the deletion operations.

3.1 Scoring scheme

Let T_1 and T_1' be the isomorphic trees which result from performing a series of deletion operations on T and T'. The isomorphism $\Phi : T_1 \to T_1'$ implies an alignment $\mathcal{M}[T,T']$ between the original trees T, T'. Let L_1, L_1' denote the sets of leaves that are mapped in T and T', respectively; because the mapping is a bijection, we must have $|L_1| = |L_1'|$. We write $\mathrm{SP} := \{(l,l') \,|\, l \in L, l' \in L', (l,l') \in M\} \subset \mathcal{M}[T,T']$ for the set of mapped pairs (we require that mapped leaves have identical labels hence the naming SP for 'species').

Recall that a mapping of two trees may involve deleting internal vertices or entire subtrees. We now distinguish between two types of internal vertex deletions.

(1) [Isolated Deletion:] deletion of only one child v of a vertex u. Let further x_1, x_2 be the two children of v. Isolated deletion of v also

implies to also delete edges $(u,v),(v,x_1),(v,x_2)$ and create new edges $(u,x_1),(u,x_2)$. Note that after deletion v has three children.

(2) [Parallel Deletion:] deletion of both children (say x and y) of a vertex v. This implies deletion and creation of edges in a fashion analogous to that for isolated deletion. Note that after deletion v has four children.

Accordingly, we further distinguish between isolated deleted vertices $N_{I,\text{iso}}$ and vertices which became deleted in parallel $N_{I,\text{par}}$ such that $N_I = N_{I,\text{iso}} \dot\cup N_{I,\text{par}}$. The idea behind distinguishing between isolated and parallel deletion is that parallel deletion reflects greater perturbation of tree topology at the same evolutionary point in time, and is less likely to occur. For a given mapping $\mathcal{M}[T,T']$ let $E_S(\mathcal{M}) := \{(u,v) \mid v \in N_{I,\text{iso}}\}$ be the set of edges which join isolated deleted vertices with their parents. Analogously, $E_P(\mathcal{M})$ is the set of edges that join deleted siblings with their parent. See Figure 2 for examples of isolated and parallel deletions.

Given a pair of mapped leaves $\tilde\ell_1, \tilde\ell_2 \in \text{SP}$ their alignment score, $\kappa(\tilde\ell_1,\tilde\ell_2)$ is defined as

$$\kappa(\tilde\ell_1,\tilde\ell_2) = C - |\theta(\tilde\ell_1) - \theta(\tilde\ell_2)|$$

where C is a positive constant, providing a positive contribution to the overall score because of the mapping of two leaves with the same label while we subtract the difference between the distances of $\tilde\ell_1$ and $\tilde\ell_2$ from the root for penalizing the mapping between two leaves which have topologic differences.

The total score $\mathcal{S}$ of an alignment $\mathcal{M}[T,T']$ as per the above definition is fully specified by

$$\begin{aligned}\mathcal{S}(\mathcal{M}[T,T']) = &\sum_{(\tilde\ell_1,\tilde\ell_2)\in\text{SP}} \kappa(\tilde\ell_1,\tilde\ell_2) \\ &- \sum_{e_s \in E_{\text{iso}}(\mathcal{M})} E\cdot w(e_s) - \sum_{e_p \in E_{\text{par}}(\mathcal{M})} F\cdot w(e_p)\end{aligned} \tag{4}$$

where, with respect to the formulation in (3), the term in the first row is for $\sum_{v,v'\in M} S_M(v,v')$, the second row is for $\sum_{v\in N_I} S_{N_I}(v)$ and $\sum_{v\in N_T} S_{N_T}(v)$ is zero. E and F are user-defined constants that, respectively, penalize *isolated deletion* and *parallel deletion* of edges. Note that this penalty is proportional to the length of the edges joining the deleted vertices with their parents—deletion of longer edges leads to a more severe perturbation of topology hence is more severely penalized. We set the cost of deleting a subtree (i.e. S_{N_T}) to 0. Note, however, deleting subtrees is implicitly penalized by disregarding any potential good mappings of leaves in them.

Given the above score function, the *gene tree alignment problem* can be formally stated as follows.

Gene tree alignment problem. Given two rooted weighted-edge trees T,T', determine subsets of leaves $L_1 \subset L, L_1' \subset L'$ of equal size such that the corresponding subtrees can be transformed by isolated and parallel deletion and subtree removal operations into trees T_1, T_1', for which there is an isomorphism $\Phi : T_1 \to T_1'$ that maximizes $\mathcal{S}(\mathcal{M}[T,T'])$.

3.2 A dynamic programming solution

The gene tree alignment problem can be efficiently solved by a dynamic programming algorithm. Our algorithm runs in $O(|V|\cdot|V'|)$ time for two binary, rooted trees T,T' with vertex sets V,V'. In general, our strategy can be applied to arbitrary rooted trees with bounded maximum degree, $\Delta_{\max}$. Note that by allowing to delete internal vertices (i.e. contract the edges), the number of children of an internal vertex will be still bounded by a constant ($\leq$4).

Initialization. As a first step, we remove all leaves that refer to species that are unique to each tree. Let $n = |V|$ and $n' = |V'|$. For every pair of vertices $v_i \in V$ and $v_j' \in V'$ (i.e. for every $i = 1,\ldots,n$ and $j = 1,\ldots,n'$), we compute the maximum alignment score for the subtrees rooted at v_i from T [i.e. $T(v_i)$] and v_j' from T [i.e. $T'(v_j')$]. We denote the maximum alignment score for $T(v_i)$ and $T'(v_j')$ by S_{ij}. Note that the computation of the maximum alignment score between rooted subtrees induce a mapping between their leaves.

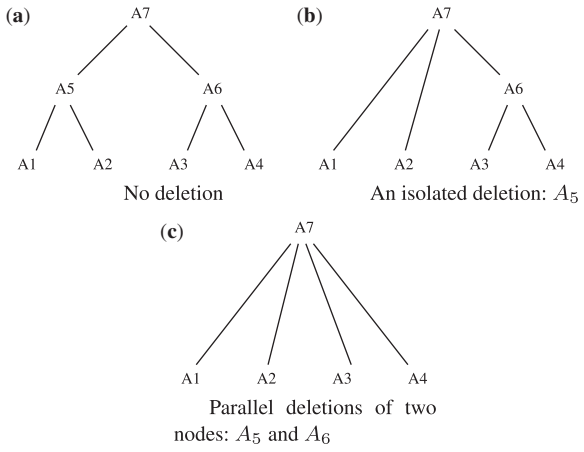

Fig. 2. A gene tree (**a**), with an isolated node deletion, A_5 (**b**) and a parallel deletion of the nodes, A_5 and A_6 (**c**).

In our dynamic programming algorithm, we handle the 'base' cases, where one (or both) of $T(v_i)$ or $T(v_j')$ have three or fewer leaves, as follows.

- If both $v_i \in V$ and $v_j' \in V'$ are leaves, then by definition, $S_{ij} = \kappa(v_i, v_j')$.
- Without loss of generality, if v_i is a leaf and v_j' is an internal vertex, $S_{ij} = \max(S_{ij_1}, S_{ij_2})$, where j_1 and j_2 correspond to the children of v_j'.
- The remainder of the base cases have both v_i and v_j as internal vertices and are solved through exhaustive evaluation of all possible alignments.

Recursion. Internal vertices, each with at least four descendants, S_{ij} will be computed through recurrence equations. These equations are based on the alignment scores between subtrees rooted at the children (or grandchildren) of v_i and v_j'. Let $i_1(j_1)$ and $i_2(j_2)$ be the children of the vertex $v_i(v_j')$. Also, let i_{11}, i_{12} be the children of i_1, and i_{21}, i_{22} be the children of i_2. Similarly, let j_{11}, j_{12} be the children of j_1, and j_{21}, j_{22} be the children of j_2. We first give a high-level description of the recurrence equation. Suppose that the maximum alignment score between any subtree in $T(v_i)$ and any subtree in $T'(v_j')$ has already been computed. In order to compute the alignment score S_{ij}, we consider several cases: we can either delete one or both subtrees rooted at the children of v_i and v_j' (deleting an entire subtree) or align the subtrees rooted at the children of v_i and v_j' to each other. We can also delete one of the children of v_i (either i_1 or i_2) together with one of the children of v_j (either j_1 or j_2) and align the three resulting subtrees in $T(v_i)$ to a *permutation*[2] of the ones in $T'(v_j')$. Finally, we have to consider the case where both children of the root [i.e. i_1 and i_2 in $T(v_i)$, and j_1 and j_2 in $T'(v_j')$] are deleted. In this case, we align four subtrees in $T(v_i)$ (rooted at $i_{11}, i_{12}, i_{21}, i_{22}$) to a permutation of the four resulting subtrees in $T'(v_j')$. The optimal alignment score of S_{ij} will thus be the *maximum* alignment score provided by all of the cases above.

Let $e(v)$ denote the penalty for isolated deletion of an internal vertex v, which is the product of the constant E and the weight of the edge between v and its parent (see Section 3.1). Also, let $f(v)$ denote the penalty for parallel deletion of both children of an internal vertex v. $f(v)$ was defined as a constant

[2]We have to consider all the permutations because the trees are unordered (i.e. the order of siblings of an internal vertex is unimportant).

F times the total weight of the edges that connect v to its children. The recurrence equation for S_{ij} thus becomes the following

$$S_{ij} = \max \begin{cases} 0 \text{ (deleting both subtrees from each tree)} \\ \left\{ \begin{matrix} S_{i_1 j_1} + S_{i_2 j_2} \\ S_{i_1 j_2} + S_{i_2 j_1} \end{matrix} \right\} \text{regular cases} \\ \left\{ \begin{matrix} S_{ij_1} \\ S_{ij_2} \\ S_{i_1 j} \\ S_{i_2 j} \end{matrix} \right\} \text{deleting one subtree from each tree} \\ \max \left\{ \begin{matrix} S_{i_{11} j_2} + S_{i_{12} j_{11}} + S_{i_2 j_{12}} \\ S_{i_{11} j_2} + S_{i_{12} j_{12}} + S_{i_2 j_{11}} \\ S_{i_{12} j_2} + S_{i_{11} j_{11}} + S_{i_2 j_{12}} \\ S_{i_{12} j_2} + S_{i_{11} j_{12}} + S_{i_2 j_{11}} \end{matrix} \right\} - e(i_1) - e(j_1) \\ \max \left\{ \begin{matrix} S_{i_{21} j_2} + S_{i_{22} j_{11}} + S_{i_1 j_{12}} \\ S_{i_{21} j_2} + S^{i_{22} j_{12}} + S_{i_1 j_{11}} \\ Si_{22} j_2 + S^{i_{21} j_{11}} + S_{i_1 j_{12}} \\ S_{i_{22} j_2} + S_{i_{21} j_{12}} + S_{i_1 j_{11}} \end{matrix} \right\} - e(i_2) - e(j_1) \\ \max \left\{ \begin{matrix} S_{i_{21} j_1} + S_{i_{22} j_{21}} + S_{i_1 j_{22}} \\ S_{i_{21} j_1} + S_{i_{22} j_{22}} + S_{i_1 j_{21}} \\ S_{i_{22} j_1} + S_{i_{21} j_{21}} + S_{i_1 j_{22}} \\ S_{i_{22} j_1} + S_{i_{21} j_{22}} + S_{i_1 j_{21}} \end{matrix} \right\} - e(i_2) - e(j_2) \\ \max \left\{ \begin{matrix} S_{i_{11} j_1} + S_{i_{12} j_{21}} + S_{i_2 j_{22}} \\ S_{i_{11} j_1} + S_{i_{12} j_{22}} + S_{i_2 j_{21}} \\ S_{i_{12} j_1} + S_{i_{11} j_{21}} + S_{i_2 j_{22}} \\ S_{i_{12} j_1} + S_{i_{11} j_{22}} + S_{i_2 j_{21}} \end{matrix} \right\} - e(i_1) - e(j_2) \\ S_{i_{11}\pi_1} + S_{i_{12}\pi_2} + S_{i_{21}\pi_3} + S_{i_{22}\pi_4} - f(v_i) - f(v_j) \end{cases} \quad (5)$$

where the *permutation* $\pi = \pi_1\pi_2\pi_3\pi_4$ ranges over all permutations of $\{j_{11}, j_{12}, j_{21}, j_{22}\}$. Note that some cases are redundant but are still represented here for the sake of clarity. In case that several options yield the same, optimal score, the algorithm picks the first observed one.

Now, given r and r', the roots of T and T', respectively, the alignment score $S_{r_1 r_2}$ (i.e. the maximum alignment score of the rooted trees) can be computed using the above recurrence equation, providing a solution to the gene tree alignment problem. It is quite straightforward to prove that our algorithm correctly computes the maximum alignment score through a (strong) induction on the *sum* of the heights of the rooted trees. Note that the scores of internal vertex alignments can be computed through the scores of the alignments between their (grand)children and the recurrence precisely serves to satisfy the constraints. The base of the induction is trivial. If the minimum height of the trees is zero (i.e. one of the trees is just a single leaf), the optimal value of the alignment can be found using the definitions and simple case analysis. Given the subtrees $T(v_i)$ and $T'(v'_j)$, with heights h and h', respectively, we assume the induction hypothesis, that for all pairs of subtrees $T(v_p)$ and $T'(v'_q)$ with heights h_p and h_q such that $h_p + h_q < h + h'$. It is easy to verify by case analysis that all cases in the recurrence equation will be reduced to a case in which the sum of the heights of the aligned (grand) children will be less.

Evaluation criteria. We determine the maximum number of members of the two protein domain families under consideration that can be paired by following (Izarzugaza *et al.*, 2008): for each species we determine the paralogous members of the domain in the two trees that can be paired with one another (that is both members reside in the same protein) and determine the maximum number of pairs that can result from the respective potential pairings. Summing up these numbers yields the maximal size of a *correct* mapping. By the usual conventions, we denote this value as P. In other words, P is the size of the correct pairing. Among the P many potential correct pairs, we determine the ones which were inferred by the algorithm in use and refer to them as 'true positives', TP. Similarly, the number of domain pairs computed, where the respective members are not from the same protein, is referred to as 'false positives', FP. Recall (Sensitivity) is defined as Rec=TP/P and Precision (positive prediction rate) is defined as Prec=TP/(TP+FP). Note that Recall is referred to as accuracy in (Izarzugaza *et al.*, 2008). We determine Precision and Recall for each pair of trees individually. Values displayed in Figure 3, Table 1 and Figure 4 (see Section 4) are average values for all 488 co-evolving tree pairs respectively for all tree pairs satisfying the respective criteria.

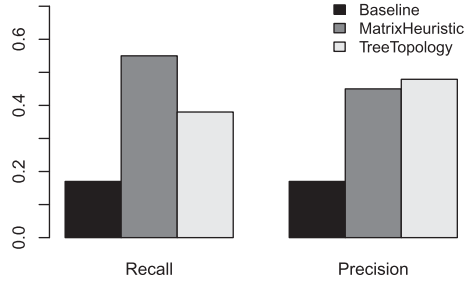

Fig. 3. Recall and Precision for the heuristic matrix-based approach [MatrixHeuristic, (Izarzugaza *et al.*, 2008)] and the deterministic, tree-topology-based approach (TreeTopology $= C_{\text{opt}}$). Baseline is determined as randomly pairing as many protein domain family members as possible. Runtimes for MatrixHeuristic and TreeTopology are 730 h and 1 min, respectively.

Table 1. Evaluation of different schoring schemes. C_{Opt} is 'TreeTopology' in the other figures.

Method	RP	Recall	RelRec	Precis
C_0	0.546	0.330	0.557	0.447
C_1	0.610	0.378	0.586	0.475
$C_{\{1,2\}}$	0.612	0.377	0.581	0.471
C_{ser}	0.638	0.373	0.556	0.444
C_{Opt}	0.612	0.380	0.588	0.479

4 RESULTS

Data source and alternative methods. We benchmarked our algorithm against the most recent heuristic search method (Izarzugaza *et al.*, 2008) for determining a mapping in the presence of paralogs on the large-scale data corpus described in the same study. This data set contains multiple alignments for 604 yeast protein domains among which 488 domain pairs are known to co-occur in the same protein. Those 488 domain family pairs are considered to be a particularly tough test (Izarzugaza *et al.*, 2008) due to the presence of ~6 paralogs per species on average. For all interacting domain family pairs, neighbor-joining trees were computed, using ClustalW (Thompson *et al.*, 1994) and the trees were rooted at the domains which are known to interact.

Tree constraints. In order to appropriately assess the contribution of the different tree constraints as outlined in Section 3, we evaluated our algorithm by not allowing to delete internal nodes (C_0), allowing isolated node deletion (C_1) as well as further allowing parallel deletion of two sibling nodes ($C_{1,2}$), see Figure 2c for an example. We also include the test case C_{ser} where we allow for deletion of a parent and a child, simultaneously. In all the above cases, we

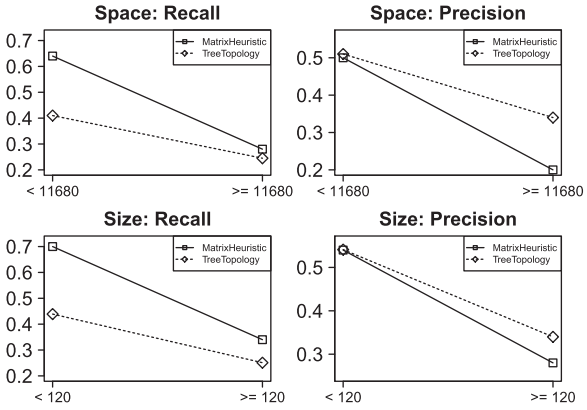

Fig. 4. The comparison of our method with the heuristic search method reveals favorable results for large trees (bottom row), *x*-axis indicates the size (number of leaves) of the larger tree of the two trees paired and in particular for large search spaces, that is for Space $\geq$11 680 where Space is the product of the number of leaves of the two trees paired.

allow deletion of subtrees as a whole without penalty. The specific scores for these cases are as follows: $C_0 : C=1, E=\infty, F=\infty$, $C_1 : C=1, E=0, F=\infty$ and $C_{1,2} : C=1, E=F=0$.

Among the cases above $C_{1,2}$ gives the best results (see Table 1 and further comments below), implying that parallel deletions are beneficial. We experimented with several values of E and F, and noticed that it may be beneficial to impose a large penalty for parallel deletions in contrast to a relatively small penalty for isolated deletions. We concluded that an optimal choice of parameters would be $E=2, F=50$ (referred to as C_{Opt}), when $C=1$. Note that the exact value of C is the function of neighbor-joining trees resulting from ClustalW multiple alignments alone—for different settings absolute values need to be put into perspective with orders of magnitude of edge weights of the trees under consideration.

As outlined in Section 3, inducing tree constraints considerably reduces the search space, thereby allowing for an efficient and deterministic method. Given a pair of trees, let CP (constraint positives) denote the maximum number of *correctly* paired domains over all possible *alignments* of the trees. Note that one can compute, CP for any given pair of trees, by running our algorithm with a scoring scheme which assigns 1 to correctly paired domains and not penalizing any operation which the constraints allow us to do.[3] We compute $RP = \text{CP}/P$ (relative positives) as the fraction of pairings that can still be inferred, and which measures the reduction of search space size due to imposing constraints. We further compute RelRec = TP/CP (Relative Recall) as a recall value which reflects how many of the correct pairings possible were inferred by the algorithm in question. The good RelRec values the tree topology approach achieves (0.59 versus 0.55 for the matrix-based approach, note that for the matrix-based approach this coincides with Recall since it does not impose any constraints), indicate that losses in Recall are due to imposing constraints, but not necessarily due to the scoring scheme.

Figure 3 presents numbers of all 488 tree pairs for a canonical baseline procedure, which randomly pairs as many domain family members per species as possible, the heuristic matrix-based approach (MatrixHeuristic) and the deterministic tree-topology-based approach (TreeTopology $= C_{\text{opt}}$). Table 1 furthermore presents numbers resulting from usage of different tree constraints. Following (Izarzugaza *et al.*, 2008), we also separate tree pairs according to the numbers of leaves of the larger tree and the product of the numbers of leaves of the two paired trees which, according to (Izarzugaza *et al.*, 2008), quantifies search space size. See Figure 4 for the respective results.

5 DISCUSSION

Runtime. The possibly most striking advantage of our topology-based approach is the drastic reduction of runtime—we can compute mappings for the 488 interacting domain families in roughly 1 min on a single CPU—in comparison to 730 h on MareNostrum[4] needed for the Metropolis search performed by (Izarzugaza *et al.*, 2008). Note that there are rapidly growing large-scale phylogenetic databases such as ENSEMBL (`http://ensembl.org`) or PhylomeDB (`http://phylomedb.org`), whose growth is further accelerated by next-generation sequencing projects (as of 12th August, 2011, PhylomeDB contains 482274 phylogenetic trees). The reduction in runtime delivered by our approach certainly overcomes a major obstacle—we render large-scale mapping and, as a consequence, comparison of paralog-rich gene trees feasible. Note that this reduction has become possible by imposing both computationally and biologically reasonable constraints on the search space while at the same time allowing for an efficient scheme to find the global optimum within these constraints.

Search space size/recall. Comparing C_{Opt} with the method of (Izarzugaza *et al.*, 2008) (heuristic) overall, clearly, Izarzugaza *et al.* (2008) achieve best recall. As pointed out above, this comes as no surprise since we cannot explore pairings that contradict the topologies of the paired trees. Quite surprisingly though, although usage of tree topology and neighbor-joining trees in particular have been discussed rather controversially (Waddell *et al.*, 2007), we find that still the majority of pairings (54.6% with the strictest constraints and 61.2% for allowing isolated and parallel deletion) can be determined by a topology-based approach. These numbers may put usage of neighbor-joining tree topology in mirrortree approaches into a general perspective. Moreover, note that the fraction of correct domain pairs computed by our method over that of the heuristic search method is $\sim 0.7 (= \text{TP}(C_{\text{opt}})/\text{TP}(\text{heuristic}) = \text{Recall}(C_{\text{opt}})/\text{Recall}(\text{heuristic}) = \frac{0.38}{0.55})$ which is more than what was to be expected by reduction of the search space $(CP(C_{opt})/CP(\text{heuristic}) = \text{CP}(C_{\text{opt}})/P(\text{heuristic}) = \text{RP}(C_{\text{opt}}) = 0.61)$ which points out that we compensate search space reduction by a more effective search strategy. This becomes reflected by the better RelRec values of C_{opt}.

Precision. Precision favors the topology-based approach, at least on larger (combinations of) trees (see figure 4 and table 1). Better precision reflects a larger fraction of the correct domain pairs among the pairs inferred overall. We achieve slightly better values in terms

[3]This scoring scheme assumes knowledge we are not allowed to use in the algorithm; we use this knowledge for the purpose of evaluation here.

[4]MareNostrum is a supercomputer of the Barcelona Supercomputing Center, one of the largest machines in the world dedicated to science (Izarzugaza *et al.*, 2008, p. 10).

of Precision than (Izarzugaza *et al.*, 2008), see Precision in Figure 3. See also Figure 4 for a comparison with respect to search space size-related differences. While Izarzugaza *et al.* (2008) achieve excellent values on pairs of smaller trees, we outperform their approach on larger trees, with the most obvious differences on pairs of trees where the product of the numbers of leaves is large.

Conclusion. In summary, we have, for the first time, devised a deterministic and efficient, polynomial-runtime mirrortree approach which directly compares the gene trees, and not the distance matrices behind or giving rise to them. We have juxtaposed our approach with the most recent, state-of-the-art matrix-based heuristic search procedure without introducing further experimental biases. Most importantly, our tree topology-based algorithm lists efficiency as its decisive benefit. While recall is better for the heuristic search obviously due to that it does not impose any constraints on the search space, we only incur relatively mild losses. We achieve better results in precision, in particular when both of the mirrored trees are large. This leads us to the overall conclusion that the heuristic method remains the better choice for smaller trees and when runtime is not an issue. In case of larger trees and in particular for large-scale studies, our approach has considerable benefits. Note finally that we have been comparing neighbor-joining trees which have been repeatedly exposed as suboptimal choices of phylogenetic trees. We believe that our approach can gain from improvements in tree quality significantly more than the matrix-based approaches. Note finally that mapping domains can lead to ambiguous results due to that several homologous copies can co-occur in one protein. To resolve such issues is interesting future work.

Funding: I.H. was supported by an Natural Sciences and Engineering Research Council of Canada (NSERC) Alexander Graham Bell Canada Graduate Scholarships. A.S. was supported by a post-doctoral fellowship of the Pacific Institute of Mathematical Sciences. S.C.S. receives funds from the NSERC and Bioinformatics for Combating Infectious Diseases.

Conflict of Interest: none declared.

REFERENCES

Bille,P. (2003) Tree edit distance, alignment and inclusion. *Technical Report TR-2003-23*. IT University of Copenhagen. Available at http://itu.dk/en/Forskning/Technical-Reports/2003/Tree-Edit-Distance-Alignment-Distance-an.

Gertz,J. *et al*. (2003) Inferring protein interactions from phylogenetic distance matrices. *Bioinformatics*, **19**, 2039–2045.

Izarzugaza,J. *et al*. (2008) Enhancing the prediction of protein pairings between interacting families using orthology information. *BMC Bioinformatics*, **9**, 35.

Jothi,R. *et al*. (2005) Predicting protein-protein interaction by searching evolutionary tree automorphism space. *Bioinformatics*, **21**(Suppl. 1), i241–i250.

Lovell,S. and Robertson,D. (2010) An integrated view of molecular co-evolution in protein-protein interactions. *Mol. Biol. Evol.*, **27**, 2567–2575.

Page,R. (1994) Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.*, **43**, 58–77.

Pazos,F. and Valencia,A. (2001) Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Eng.*, **14**, 609–614.

Pazos,F. and Valencia,A. (2008) Protein co-evolution, co-adaptation and interactions. *EMBO J.*, **27**, 2648–2655.

Pellegrini,M. *et al*. (1999) Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc. Natl Acad. Sci. USA*, **96**, 4285–4288.

Pinter,R. *et al*. (2008) Approximate labelled subtree homeomorphism. *J. Discr. Algor.*, **6**, 480–496.

Ramani,A. and Marcotte,E. (2003) Exploiting the co-evolution of interacting proteins to discover interaction specificity. *J. Mol. Biol.*, **327**, 273–284.

Raynal,B. *et al*. (2010) Homeomorphic alignment of weighted trees. *Pattern Recogn.*, **43**, 2937–2949.

Tai,K.-C. (1979) The tree-to-tree correction problem. *J. ACM*, **26**, 422–433.

Thompson,J. *et al*. (1994) Clustal w: improving the sensitivity of progressive multiple sequence alignments through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **18**, 4673–4680.

Tillier,E. and Charlebois,R. (2009) The human protein coevolution network. *Genome Res.*, **19**, 1861–1871.

Tillier,E. *et al*. (2006) Codep: maximizing co-evolutionary interdependencies to discover interacting proteins. *Prot. Struc. Func. Bioinform.*, **63**, 822–831.

Vyugin,V. *et al*. (2002) Tree conciliation: reconstruction of species phylogeny by phylogenetic gene trees. *Mol. Biol.*, **36**, 650–658.

Waddell,P. *et al*. (2007) Phylogenetic methodology for detecting protein interactions. *Mol. Biol. Evol.*, **24**, 650–659.

Zhang,K. and Jiang,T. (1994) Some MAX-SNP-hard results concerning unordered labeled trees. *Inform. Process. Lett.*, **49**, 249–254.

Zhang,K. *et al*. (1992) On the editing distance between unordered labeled trees. *Inform. Process. Lett.*, **42**, 133–139.

Zhang,K. (1996) A constrained edit distance between unordered labeled trees. *Algorithmica*, **15**, 205–222.