Efficient Local Search for Euclidean Path-Difference Median Trees

Alexey Markin and Oliver Eulenstein

Abstract—Synthesizing large-scale phylogenetic trees is a fundamental problem in evolutionary biology. Median tree problems have evolved as a powerful tool to reconstruct such trees. Such problems seek a median tree for a given collection of input trees under some problem-specific tree distance. There has been an increased interest in the median tree problem for the classical path-difference distance between trees. While this problem is NP-hard, standard local search heuristics have been described that are based on solving a local search problem exactly. For a more effective heuristic we devise a time efficient algorithm for the local search problem that improves on the best-known solution by a factor of n, where n is the size of the input trees. Furthermore, we introduce a novel hybrid version of the standard local search that is exploiting our new algorithm for a more refined heuristic search. Finally, we demonstrate the performance of our hybrid heuristic in a comparative study with other commonly used methods that synthesize species trees using published empirical data sets.

Index Terms—Phylogenetic trees, median trees, supertrees, path-difference distance, local search.

1 Introduction

LARGE-scale phylogenetic trees that represent the evolutionary relationships, or genealogy, among thousands of species offer enormous promise for society's advancements. While such species trees are fundamental to evolutionary biology, they are also benefiting many other disciplines, such as agronomy, biochemistry, conservation biology, epidemiology, environmental sciences, genetics, genomics, medical sciences, microbiology, and molecular biology [1], [2], [3], [4]. However, despite these promises, synthesizing large-scale species trees is confronting us with one of the most difficult computational challenges in evolutionary biology today. Here, we are focusing on synthesizing large species trees from a given collection of typically smaller phylogenetic trees.

Traditionally, a species tree for a set of species is inferred by first selecting a gene that is common to them, and then inferring the evolutionary history for this gene, which is called a *gene tree*. Gene trees describe partial evolutionary histories of the species genomes, and therefore, it is often assumed that gene trees have evolved along the edges of the species tree, imitating it. However, a major shortcoming of the traditional approach is that different gene trees for the same set of species can describe discordant evolutionary histories. Such discordance is frequently caused by erroneous gene trees, or can be the result of genes which have evolved differently due to complex evolutionary processes that have shaped the species genomes [5]. To confront these challenges, median tree problems (also called supertree problems [6]) have emerged as a powerful tool for inferring species trees from a collection of discordant gene trees. These problems seek a tree, called a *median tree*, that is minimizing the overall distance to the input trees based on some problem-specific distance measure. Typically, measures that have been well-established in comparative phylogenetics are used to compute median trees [6], and one of the oldest measures to compare trees is the path-difference distance. However, despite the tradition and popularity of the path-difference distance, the analysis and computation of median trees under this measure are still in their infancy.

In this work we are studying the computation of path-difference median trees. Recently it has been shown that computing such median trees is an NP-hard problem for rooted as well as unrooted input trees [7] [1]. While most median tree problems used in practice are NP-hard, they have been effectively addressed by standard local search heuristics that compute exact solutions to a local search problem thousands of times. Encouraged by these promising results we introduce a novel hybrid heuristic that is based on local search to compute median trees under the path-difference distance. The heuristic is based on our $\Theta(kn^2)$ time algorithm (introduced here) that solves the corresponding local search problem exactly, where n and k are the size and number of trees in a given instance of the median tree problem respectively. This algorithm improves by a factor of n on the previously best-known algorithm [7]. Our new hybrid heuristic is exploiting the speed-up of our algorithm, allowing to compute more accurate and truly large-scale median trees under the path-difference distance. Finally, we demonstrate the performance of our new heuristic in a comparative study on several published empirical data sets, and demonstrate that it outperforms previous standard heuristics in minimizing the overall path-difference in scalability and accuracy. Software implementing our local search heuristic is freely available from the web-page

• *A. Markin and O. Eulenstein are with the Department of Computer Science, Iowa State University, Ames, IA, 50011.*
E-mail: {amarkin, oeulenst}@iastate.edu

A slightly different version of the manuscript is published and copyrighted by IEEE; DOI: 10.1109/TCBB.2017.2763137

1. Note, the work [7] is the original published conference abstract on which the presented journal version is largely extending and improving upon.

http://genome.cs.iastate.edu/ComBio/software.htm.

1.1 Related work

Median tree problems are a popular tool to synthesize large-scale species trees from a collection of smaller trees. Given a collection of input trees, such problems seek a tree, called a *median tree*, that minimizes the sum of its distances to each of the input trees. Since the ultimate goal of median tree problems is to synthesize accurately species trees of enormous scale, a large body of work has focused on the biological, mathematical, and algorithmic properties of median tree problems adopting numerous definitions of distance measures from comparative phylogenetics [6].

One of the oldest such measures, however, is the path-difference distance [8], [9], [10], [11]. The *path-difference distance* between two trees is defined through the Euclidean distance between their *path-length vectors*. Each such vector represents the pairwise distances between all leaves of the corresponding tree (i.e., the number of edges on a simple path between leaves). Steel and Penny [9] have studied the distribution of the path-difference distance for unrooted trees. Complementing this work, Mir and Rosello [12] computed the mean value of this distance for fully resolved unrooted trees with n leaves, and showed that this mean value is in $O(n^3)$. Variants of the path-difference distance are the Manhattan distance of the path-length vectors [13], and their correlation [14].

The median tree problem for the path-difference distance has been introduced only recently [7], where its NP-hardness was shown for rooted and unrooted input trees. This is due to the fact that the rooted and the unrooted version of this problem contain the *maximum compatible subset of rooted triplets problem* and the *quartet compatibility problem* as special cases respectively.

Median tree problems are in general NP-hard [6], and therefore are, in practice, approached by using local search heuristics [15], [16], [17], [18], [19] that make truly large-scale phylogenetic analyses feasible [15], [16]. Effective local search heuristics have been proposed and analyzed [15], [16], [17], [18], [19], and provided various credible species trees [15], [16]. Such heuristics have typically two phases, where phase I is constructing a suboptimal candidate tree, called a *seed tree*, on which phase II is improving on. We are describing these phases for an instance I with an overall taxon set T of a median tree problem with some minimization cost d.

Phase I is stepwise adding taxa to an initial rooted and full binary tree until the tree contains all of the taxa in T. The initial tree is a three-taxon tree with the minimum cost d to instance I under all possible three-taxon trees over T, which is found by complete enumeration. Subsequently, new trees are built by adding one of the remaining taxa of T to each of the branches in the currently-built tree by selecting one such tree with the minimum cost d when compared to I as the next built tree. The *cost* d of a candidate tree over a subset T' of T to instance I is its cost to the trees displayed by the input trees of I for the taxa in T'.

Phase II is initialized with the seed tree form phase I and finds a minimum cost tree for I under d in its (local) neighborhood, and so on, until a local minima is reached. At each local search step phase II is solving an instance of a local search problem. The time complexity of this local search problem depends on the tree edit operation that defines the neighborhood, as well as on the computation time of the tree distance measure that is used.

A classical and well-studied tree edit operation is the *subtree prune and regraft (SPR)* operation [20] where a subtree of the edited tree is pruned and regrafted back into the tree at another location. The *SPR neighborhood* of T is the set of all trees into which T can be transformed by one SPR operation, and this neighborhood contains $\Theta(n^2)$ trees. Further, the best-known algorithm to compute the path-difference distance between two trees with n leaves requires $\Theta(n^2)$ time [9].

Therefore, given an instance of k trees over n different taxa of the SPR based local search problem, this problem can be naïvely solved by complete enumeration in $\Theta(kn^4)$ time. While an $\Theta(kn^3)$ time algorithm was introduced that allowed to infer the first path-difference median tree estimates for larger empirical data sets [7], its cubic runtime is still prohibitive for synthesizing truly large-scale empirical studies.

1.2 Our Contribution

We introduce a powerful hybrid heuristic for the *path-difference median tree problem* under the classical path-difference distance to synthesize large-scale phylogenetic trees. The key component of this heuristic is an algorithm that is speeding up phase II of the standard local search heuristic by solving the SPR based local search problem in $\Theta(kn^2)$ time, where n and k is the size and number of the input trees of the median tree problem respectively. That way our algorithm improves on the previously best-known solution of $\Theta(kn^3)$ by a factor of n, which is possible by exploiting novel optimal substructures of the local search problem, leading to a more efficient dynamic programming solution. In particular, our algorithm is providing a substantial speed-up of phase II for larger instances where typically thousands of local searches have to be computed.

The previously best-known local search algorithm relies on the SPR-neighborhood *semistructure* initially introduced in [21], which enforced certain limitations on the analysis. In this work we show that the *semistructure* is not required. Without the *semistructure* limitation we were able to devise an efficient dynamic approach method for traversing the SPR-neighborhood of a candidate median tree and calculate its path-difference distance to the input trees for each tree in this neighborhood. In addition, we developed a more elaborate precomputation strategy that enables us to answer certain sum-related queries to a path-difference matrix in constant time, which is used as a subroutine in our method.

The hybrid heuristic is making excessive use of our efficient computation of phase II by interleaving phase I and phase II of the standard local search heuristic, allowing for a much more refined local search. Initially the three-taxon tree of phase I is computed, which is then used as the seed tree for phase II. The tree resulting from phase II is then used as the current-built tree for the stepwise taxon addition of phase I. The resulting tree is again the seed tree for phase II, and so on, until phase II is executed on a seed tree having

all of the taxa. Thus, the hybrid heuristic is using our new efficient local search algorithm to optimize on the current-built tree on every step of the taxa addition of phase I.

In our experimental evaluation we present a scalability analysis that compares our novel algorithm with the previously best-known solution on simulated phylogenetic data. Additionally, we demonstrate the performance of our new local search heuristic through comparative studies using empirical data sets and evaluate the effectiveness of the presented hybrid heuristic. In this study we compare our hybrid heuristic with the standard two-phase heuristic and other widely recognized supertree methods. Finally, to more accurately relate the empirical results, we estimate path-difference distance distributions for empirical data sets and map distances of supertrees obtained via standard methods under consideration on them.

2 Basics and Preliminaries

Basic definitions. A *(phylogenetic) tree* T is a rooted full binary tree. We denote its node set, edge set, leaf set, and root, by $V(T)$, $E(T)$, $\mathsf{L}(T)$, and $\mathsf{Rt}(T)$ respectively. Given a node $v \in V(T)$, we denote its parent by $\mathsf{Pa}_T(v)$, its set of children by $\mathsf{Ch}_T(v)$, its sibling by $\mathsf{Sb}_T(v)$, the subtree of T rooted at v by $T(v)$, and $T|v$ is the phylogenetic tree that is obtained by pruning $T(v)$ from T. Note that we identify the leaf set of a phylogenetic tree with the respective set of leaf-labels (taxa).

Let $L \subseteq \mathsf{L}(T)$ and T' be the minimal subtree of T with leaf set L. We define the *leaf-induced subtree* $T[L]$ of T to be the tree obtained from T' by successively removing each node of degree two (except for the root) and adjoining its two neighbors.

Additionally, for any node $v \in V(T)$ we let C_v to be the *cluster* of the node v, which is defined as $C_v := \mathsf{L}(T(v))$ (i.e., a set of taxa in the subtree of T rooted at v).

Path-difference distance. Given a tree T and two leaves $u, v \in \mathsf{L}(T)$, let $\mathsf{d}_{u,v}(T)$ denote the length in the number of edges of the unique path between u and v in T. Let $\mathsf{d}(T)$ be an associated vector obtained by a fixed ordering of pairs i, j [9], e.g., $\mathsf{d}(T) = (\mathsf{d}_{1,2}(T), \mathsf{d}_{1,3}(T), \ldots, \mathsf{d}_{n-1,n}(T))$, where n is the number of leaves. Then the *path-difference distance (PDD)* between two trees G and S over the same leaf set is defined as

$$\mathsf{d}(G, S) := ||\, \mathsf{d}(G) - \mathsf{d}(S)||_2.$$

We also define $\mathsf{PLM}(T)$ to be the matrix of path-lengths between each two leaves in T. That is, a matrix of size $|\,\mathsf{L}(T)| \times |\,\mathsf{L}(T)|$, where rows and columns represent leaves of T, and $\mathsf{PLM}_{u,v}(T) = \mathsf{d}_{u,v}(T)$. Let G and S be trees over the same leaf set, then we define $\Delta(G, S) := \mathsf{PLM}(G) - \mathsf{PLM}(S)$ to be the *path-difference matrix*.

3 Path-difference median tree problem

Let $\mathcal{P}$ be a set of trees $\{G_1, \ldots, G_k\}$. We define $\mathsf{L}(\mathcal{P}) := \cup_{i=1}^{k} \mathsf{L}(G_i)$ to be the *leaf set of* $\mathcal{P}$. A tree S is called a *supertree* of $\mathcal{P}$, if $\mathsf{L}(S) = \mathsf{L}(\mathcal{P})$. Further, we extend the definition of the path-difference distance to a set of trees. Note, we defined PDD only for two trees over the same leaf set. However, we do not want to enforce such a restriction on the set of input trees, since typically supertrees are larger than input trees. Therefore, in order to compare two trees S and G, where $\mathsf{L}(G) \subseteq \mathsf{L}(S)$ we use the classical *minus method* [22]. That is, we calculate a distance between G and the subtree of S induced by $\mathsf{L}(G)$: $\mathsf{d}(S, G) = \mathsf{d}(S[\mathsf{L}(G)], G)$. We now define PDD for an input set $\mathcal{P}$ and a supertree S as a sum $\mathsf{d}(\mathcal{P}, S) := \sum_{i=1}^{k} \mathsf{d}(G_i, S[\mathsf{L}(G_i)])$, which is used to establish the following problem.

Problem 1 (PD median tree (supertree) – decision version).
Instance: a set of input trees $\mathcal{P}$ and a real number p;
Question: determine whether there exists a supertree S, such that $\mathsf{d}(\mathcal{P}, S) \leq p$.

PD median tree problem was shown to be NP-hard in the pioneer work on this problem [7] for both rooted and unrooted median trees. For the rooted case a reduction from the NP-hard maximum compatible subset of rooted triplets problem [23] was used. As for the unrooted case, a reduction from the celebrated quartet compatibility problem [24] was presented in [7].

4 Local search for PD median tree problem

As stated in the introduction, we address the NP-hardness by devising a new SPR based local search heuristic. Next, we introduce needed definitions.

4.1 SPR-based Local search

Definition 4.1. Given a node $v \in V(S) - \{\mathsf{Rt}(S)\}$, and a node $u \in V(S) - (V(S(v)) \cup \{\mathsf{Pa}(v)\})$, $SPR_S(v, u)$ is a tree obtained as follows:

1) Prune the subtree $S(v)$ by (i) removing the edge $\{\mathsf{Pa}(v), v\}$, and (ii) removing $\mathsf{Pa}(v)$ by adjoining its parent and child.
2) If u is a root of $S|v$, then a new root w' is introduced, so that u is a child of w'. Otherwise, an edge $(\mathsf{Pa}(u), u)$ is subdivided by a new node w'.
3) Connect the subtree $S(v)$ to the node w'.

In addition, we introduce the following useful notation

$$SPR_S(v) := \bigcup_u SPR_S(v, u);$$

$$SPR_S := \bigcup_{v,u} SPR_S(v, u).$$

SPR_S is called an *SPR-neighborhood* of a tree S, and $|SPR_S| = O(n^2)$, where $n = |\,\mathsf{L}(S)|$.

Given a set of input trees $\mathcal{P} = \{G_1, \ldots, G_k\}$, the search space in the median tree problem can be viewed as a graph $\mathcal{T}$, where nodes represent supertrees of $\mathcal{P}$. There is an edge $\{S_1, S_2\}$ in $\mathcal{T}$, if S_1 can be transformed to S_2 with a single SPR operation. As was mentioned in the introduction, local search is designed to terminate at a local minimum of $\mathcal{T}$. More formally, at each iteration the following problem is solved

Problem 2 (PD local search).
Instance: An input set $\mathcal{P}$ and a supertree S;
Find: $S' = \underset{S' \in SPR_S}{\arg\min}\ \mathsf{d}(\mathcal{P}, S')$.

Next we describe an algorithm for the PD local search problem that improves on the best-known algorithm (see Introduction) by a factor of n.

4.2 Dynamic programming approach to local search

Let $\mathbf{G} \in \mathcal{P}$ be a fixed input tree, and let S_i be a supertree in the i-th iteration of the local search. Throughout this section we refer to the restricted tree $S_i[\mathsf{L}(G)]$ as simply $\mathbf{S}$. To find structural properties for the dynamic programming approach, we explore how the PD distance changes when a single SPR operation is performed.

Let $T = SPR_S(v, w)$ and let e_w be the edge in S where the new parent of v will be placed as a result of the regrafting operation. Note that for convenience we assume that any phylogenetic tree has an auxiliary edge that goes into the root node, which we denote by $\{\infty, \mathsf{Rt}(S)\}$. That way, when w is $\mathsf{Rt}(S)$ (meaning that we regraft $S(v)$ above the root) e_w is exactly that auxiliary edge. Now let $U_T = (u_0, u_1, ..., u_t)$ be the *SPR-path* in S that starts from $u_0 = v$ and ends with the edge e_w. That is, $\{u_{t-1}, u_t\} = e_w$. This path is illustrated in Figure 1a. The figure clearly shows the separation of the tree S on disjoint subtrees along the path.

These subtrees are the focus of our further analysis. The main observation is that the distances between leaves within a single such subtree do not change, and we only need to keep track of the change in-between these subtrees. Note that in this section we talk about subtrees and corresponding leaf-clusters in an "unrooted" sense. That is, for a node u we say that $S(u)$ is a *lower subtree* of u, and $\overline{S(u)}$ is an *upper subtree* of u (basically, $\overline{S(u)} = S|u$, but the $\overline{S(u)}$ notation will become useful later on). The same applies to clusters of leaves – if there is a cluster C_u, then we also have a complementary cluster $\overline{C_u} = C_{\mathsf{Rt}(S)} - C_u$ that correspond to the leaf set of the upper subtree of u. The reason, why we need this extension is that the root of S could be either somewhere on the path or within one of the subtrees along the path. In the latter case such a subtree is an *upper* subtree of the corresponding node on the path.

Now we are ready to introduce more helpful notation. Consider any node u_i on the path, where $1 \leq i \leq t-1$. Informally, we let R_i to be the leaf set of the subtree that "comes out" of u_i as suggested in Figure 1a. To formally define R_i we consider the following two cases:

- u_{i-1} and u_{i+1} are both children of u_i. In that case $R_i = \overline{C_{u_i}}$.
- either u_{i-1} or u_{i+1} is a parent of u_i. W.l.o.g. let u_{i-1} be a child of u_i, then let p be the other child of u_i. In that case $R_i = C_p$.

In addition, we let R_t be the leaf set of the subtree that is connected to the SPR-path U_T by the edge e_w. Once again, we have two cases to consider to formally define R_t:

- u_t is the parent of u_{t-1}. Then $R_t = \overline{C_{u_{t-1}}}$. Note that R_t could be an empty set (i.e., when e_w is the auxiliary edge above the root node).
- otherwise, $R_t = C_{u_t}$.

Observe that the subtrees depicted schematically in Figure 1 account for the whole leaf set of S combined together. That is, $\mathsf{L}(S) = (\bigcup_{i=1}^{t} R_i) \cup C_v$. Table 1 shows a path-length difference matrix $\mathsf{PLM}(T) - \mathsf{PLM}(S)$. Using this table, it is possible to derive the difference between $\mathsf{d}(T, G)$ and $\mathsf{d}(S, G)$. Below we explain the table by exploring how the path between two leaves i and j changes when regrafting the node v above w.

(i) $i \in C_v$, $j \in R_t$. In S the path between i and j can be denoted by $A_i \sqcup (u_1, \ldots, u_t) \sqcup B_j$. Note that the partial paths A_i and B_j are not changed by the regrafting operation. In T the path between i and j is $A_i \sqcup (\mathsf{Pa}_T(v), u_t) \sqcup B_j$. The number of edges in the path is decreased by $t - 2$.

(ii) $i \in C_v$, $j \in R_1$. Again, we denote the path between i and j in S by $A_i \sqcup (u_1) \sqcup B_j$. Then the corresponding path in T is $A_i \sqcup (\mathsf{Pa}_T(v), u_{t-1}, \ldots, u_2) \sqcup B_j$. The path length increased by $t - 2$.

(iii) $i \in C_v$, $j \in R_p$, where $1 \leq p < t$. We denote the path between i and j in S by $A_i \sqcup (u_1, \ldots, u_p) \sqcup B_j$. Then the corresponding path in T is $A_i \sqcup (\mathsf{Pa}_T(v), u_{t-1}, \ldots, u_p) \sqcup B_j$. It is easy to see that the path length increased by $(t - p) - (p - 1)$.

(iv) $i \in R_t$, $j \in R_p$, where $1 < p < t$. We denote a path between i and j in S by $A_i \sqcup (u_t, \ldots, u_p) \sqcup B_j$. Then the corresponding path in T is $A_i \sqcup (u_t, \mathsf{Pa}_T(v), u_{t-1}, \ldots, u_p) \sqcup B_j$. Exactly one edge was added to the path.

(v) $i \in R_1$, $j \in R_p$, where $1 < p < t$. We denote a path between i and j in S by $A_i \sqcup (u_1, \ldots, u_p) \sqcup B_j$. Then the corresponding path in T is $A_i \sqcup (u_2, \ldots, u_p) \sqcup B_j$. Exactly one edge was removed from the path.

(vi) It is not difficult to check that in all other cases path-lengths are not affected by the regrafting operation.

Let A and B be two elements from $\{C_v, R_1, \ldots, R_t\}$ (set of disjoint subsets), and $dif_{A,B}$ be the corresponding value according to Table 1. For convenience we will refer to $\Delta(S, G)$ as simply Δ.

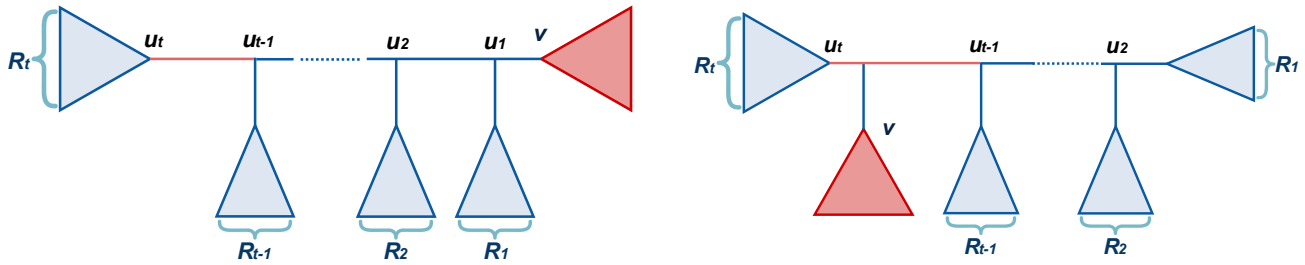
$$\begin{aligned} &\mathsf{d}^2(T, G) - \mathsf{d}^2(N, G) \\ &= \textstyle\sum_{\forall\{A,B\}} \sum_{\substack{i \in A \\ j \in B}} (\Delta_{i,j} + dif_{A,B})^2 - (\Delta_{i,j})^2 \\ &= \textstyle\sum_{\forall\{A,B\}} (|A||B| dif_{A,B}^2 + 2 dif_{A,B} \sum_{\substack{i \in A \\ j \in B}} \Delta_{i,j}). \end{aligned} \tag{1}$$

4.2.1 Precomputation

The above equation shows that in order to efficiently calculate $\mathsf{d}(T, G)$ for an arbitrary $T \in SPR_S$ we need to know $\sum_{\substack{a \in A \\ b \in B}} \Delta_{a,b}$ for every pair of distinct A, B, such that $dif_{A,B} \neq 0$. Since A and B represent leaf sets of some subtrees of S it would suffice to precompute sums over submatrices of Δ corresponding to *all* possible pairs of subtrees. Next we design an algorithm that allows us to perform such precomputation in only $O(n^2)$ time using a dynamic programming approach.

Algorithm idea. Assume that we somehow assigned indices to all subtrees of S (both lower and upper subtrees). Given an index i let S_i denote the corresponding subtree of S. In total we have $2 \cdot (2n - 1)$ such subtrees. Our goal is to fill out a matrix M of size $(4n - 2) \times (4n - 2)$, where an entry $M_{i,j}$ corresponds to a sum $\sum_{\substack{a \in \mathsf{L}(S_i) \\ b \in \mathsf{L}(S_j)}} \Delta_{a,b}$.

In order to simplify the analysis we identify subtrees and corresponding indices. That is, we let $M(S_i, S_j)$ denote



(a) The original tree S. The edge shown in red is e_w.

(b) The tree $T = SPR_S(v, w)$.

Fig. 1: The two figures schematically show the SPR operation and the way it affects the SPR-path U_T. Note that one of the subtrees, except for $S(v)$ that is depicted in red, could contain a root. Additionally, one of these subtrees could be empty.

TABLE 1: Here $1 < p < t$. Values inside the table indicate the difference in path lengths between leaves from different subsets, i.e., for $i \in C_v$ and $j \in R_1$: $\mathsf{d}_{i,j}(T) = \mathsf{d}_{i,j}(S) + t - 2$.

	$\mathbf{C_v}$	$\mathbf{R_1}$	... $\mathbf{R_p}$ ...	$\mathbf{R_t}$
$\mathbf{C_v}$	0	$t-2$	$-2p+1$ $+t$	$2-t$
$\mathbf{R_1}$	$t-2$	0	-1	0
$\vdots$				
$\mathbf{R_p}$	$-2p+1$ $+t$	-1	0	1
$\vdots$				
$\mathbf{R_t}$	$2-t$	0	1	0

$M_{i,j}$ for any two subtrees S_i and S_j, and $M(S_i)$ denote a single row M_i. To fill out the matrix M we apply a dynamic bottom-up approach. That is, for each node $u \in V(S)$ we fill out a row $M(S(u))$ starting from leaves up. Note that we are not interested in $M(S_i, S_j)$ values, when both S_i and S_j are *upper* subtrees, since there could be *no more than one* upper subtree along any SPR-path. Therefore, we explicitly compute only rows for lower subtrees (i.e., $M(S(u))$) and we do not consider upper subtree ($M(S|u)$) rows.

Base case. In the base case u is a *leaf*. Now let y be some node in $V(S)$. Then $M(S(u), S(y))$ can be equivalently computed through the following relations:

$$M(S(u), S(y)) = \sum_{c \in \mathsf{Ch}(y)} M(S(u), S(c)),$$

when y is not a leaf;

$$M(S(u), S(y)) = \Delta_{u,y}$$

otherwise.

Having $M(S(u), S(y))$ computed for all y using, once again, a bottom-up approach, we can proceed to computing the values of $M(S(u), S|y)$ for all y with a top-down strategy.

$$M(S(u), S|y) = M(S(u), S|\,\mathsf{Pa}(y)) + M(S(u), S(\mathsf{Sb}(y))),$$

when y is not the root;

$$M(S(u), S|y) = 0$$

otherwise.

Step up. Now consider any *internal* node $u \in V(S)$, whose children we denote by c_1 and c_2. Then for any fixed subtree S_j we observe the following relationship:

$$M(S(u), S_j) = M(S(c_1), S_j) + M(S(c_2), S_j)$$

Therefore, if we have the rows $M(S(c_1))$, $M(S(c_2))$ computed, we can compute the row $M(S(u))$ in time $\Theta(n)$.

Matrix computation time complexity. Computation of a row $M(S(u))$ both in the base case and not takes $\Theta(n)$ time. Therefore, to compute all such rows we need $\Theta(n^2)$ time.

W.l.o.g. assume that the subtrees along the SPR-path U_T were indexed from 0 to t. That is, the $S(v)$ subtree has index 0, the subtree with a leaf set R_1 has index 1, and so on. Then, having the matrix M we can compute the desired difference $\mathsf{d}^2(T,G) - \mathsf{d}^2(S,G)$ by using the Equation 1. Below we expand this equation and break it into components that correspond to different cells (or groups of cells) in Table 1.

$$\mathsf{d}^2(T,G) - \mathsf{d}^2(S,G) =$$

(i) Let $A = C_v$ and $B = R_t$, then $dif_{A,B} = 2 - t$. By making a substitution in the Equation 1 we get

$$|C_v||R_t|(2-t)^2 + 2(2-t)M(C_v,R_t)$$

(ii) Let $A = C_v$ and $B = R_1$, then $dif_{A,B} = t - 2$.

$$+|C_v||R_1|(t-2)^2 + 2(t-2)M(C_v,R_1)$$

(iii) Let $A = C_v$ and $B = R_i$ $(1 < i < t)$, then $dif_{A,B} = t - 2i + 1$. Summing up over all i we get

$$+ \sum_{1\le i<t} |C_v||R_i|(t-2i+1)^2 + 2(t-2i+1)M(C_v,R_i)$$

(iv) Let $A = R_1$ and $B = R_i$ $(1 < i < t)$, then $dif_{A,B} = -1$. It easy to see that instead of summing up over all i we can equivalently present this component as follows:

$$+ |R_1|(|\,\mathsf{L}(S) - C_v - R_1 - R_t|) + 2(-1)(M(R_1,\overline{R_1}) - M(C_v,R_1) - M(R_1,R_t))$$

(v) Let $A = R_t$ and $B = R_i$ $(1 < i < t)$, then $dif_{A,B} = 1$. The corresponding component can be written as follows:

$$+ |R_t|(|\,\mathsf{L}(S) - C_v - R_1 - R_t|) + 2(M(R_t,\overline{R_t}) - M(C_v,R_t) - M(R_1,R_t))$$

Note that all the components derived above can be calculated in constant time, except for item (iii) that includes a sum over the subtrees on the SPR-path, which we denote as $\mathcal{Q}(T,t)$.

$$\mathcal{Q}(T,t) = \sum_{1<i<t} \begin{pmatrix} |C_v||R_i|(t-2i+1)^2 \\ +2(t-2i+1)M(C_v,R_i) \end{pmatrix} \quad (2)$$

Next, we present an algorithm that traverses the SPR-neighborhood of S in such way that it is becomes possible to calculate the above difference in constant time for any $T \in SPR_S$.

4.2.2 Traversing the SPR-neighborhood

The time required to compute the value of $\mathcal{Q}(T,t)$ (see Equation 2) depends on the length of the SPR-path U_T. Let's consider how this value changes, when we increase the length of the SPR-path. That is, let $U_T = (u_0,\ldots,u_t)$ be the original SPR-path of a fixed tree $T = SPR(v,w)$. Now consider another tree $T' = SPR(v,w')$ whose SPR-path is by one node longer: $U_{T'} = (u_0,\ldots,u_t,u_{t+1})$. The SPR-path $U_{T'}$ goes one node "deeper" in the subtree S_t; thus, it splits S_t in two subtrees, leaf sets of which we will denote by R'_t and R'_{t+1} according to their order on the path $U_{T'}$.

It is easy to verify that the following equation captures the difference between $\mathcal{Q}(T',t+1)$ and $\mathcal{Q}(T,t)$:

$$\begin{aligned}\mathcal{Q}(T',t+1) - \mathcal{Q}(T,t) = &\sum_{1<i<t} |C_v||R_i| \\ &+ 2\sum_{1<i<t} (t-2i+1)|C_v||R_i| \\ &+ 2\sum_{1<i<t} M(C_v,R_i) \\ &+ \big(|C_v||R_{t+1}|(-(t+1)+1)^2 \\ &\quad + 2(-(t+1)+1)M(C_v,R_{t+1})\big).\end{aligned}$$

Clearly, to calculate the difference $\mathcal{Q}(T',t+1) - \mathcal{Q}(T,t)$ in constant time we need to maintain the following components:

1) $\mathrm{d}\mathcal{Q}_1(T,t) = \sum_{1<i<t} |C_v||R_i|$
2) $\mathrm{d}\mathcal{Q}_2(T,t) = \sum_{1<i<t} (t-2i+1)|C_v||R_i|$
3) $\mathrm{d}\mathcal{Q}_3(T,t) = \sum_{1<i<t} M(C_v,R_i)$

Base case. In the base case we consider an SPR-path with the smallest length, which is 3. In this case we can calculate $\mathcal{Q}$ and the values $\mathrm{d}\mathcal{Q}_1, \mathrm{d}\mathcal{Q}_2, \mathrm{d}\mathcal{Q}_3$ directly in constant time.

Increasing the SPR-path length. When we increase the SPR-path length by 1, we need to update the values of $\mathrm{d}\mathcal{Q}_1, \mathrm{d}\mathcal{Q}_2, \mathrm{d}\mathcal{Q}_3$. It is easy to observe that the following relations allow us to do that:

1) $\mathrm{d}\mathcal{Q}_1(T',t+1) = \mathrm{d}\mathcal{Q}_1(T,t) + |C_v||R'_t|$
2) $\mathrm{d}\mathcal{Q}_2(T',t+1) = \mathrm{d}\mathcal{Q}_2(T,t) + \mathrm{d}\mathcal{Q}_1(T,t) + (2-t)|C_v||R'_t|$
3) $\mathrm{d}\mathcal{Q}_3(T',t+1) = \mathrm{d}\mathcal{Q}_3(T,t) + M(C_v,R'_t)$

Sub-neighborhood traversal. We fix a node $v \in V(S)$, such that $v \neq \mathsf{Rt}(S)$. In order to apply the introduced above dynamic strategy we need to traverse the $SPR_S(v)$ sub-neighborhood so that for every new tree $T = SPR_S(v,w)$ with a corresponding path $U_T = (u_0,\ldots,u_t)$ that we consider, the path should either fall under the *base case*, or a shorter path $(u_0,\ldots,u_{t-1})$ should have been already processed. The following node *iterator* allows us to achieve that goal.

```
1: function RegraftAboveNodeIterator(Tree S, Pruned node v)
2:     Node n := Sb(v)
3:     repeat
4:         for Node w in preorder-iterator(S(n)) do
5:             YIELD w
6:         end for
7:         if n ≠ Sb(v) then
8:             YIELD Pa(n)
9:         end if
10:        n := Sb(Pa(n))
11:    until n is Null
12: end function
```

If we apply this strategy for each $v \in V(S) - \{\mathsf{Rt}(S)\}$, we obtain an algorithm that calculates the $\mathsf{d}(T,G)$ value for each $T \in SPR_S$ in constant ($O(1)$) time. Since the size of SPR_S is $\Theta(n^2)$ the overall time needed to calculate $\mathsf{d}(T,G)$ for *all* T is $\Theta(n^2)$ with a single precomputation step of time complexity $\Theta(n^2)$ as well. Next, we need to perform

this procedure for each input tree G in order to obtain the desired values of $\mathrm{d}(T, \mathcal{P})$. Therefore, the overall time complexity of a single local search iteration is $\Theta(kn^2)$.

4.3 Unrooted case

Most of the above results apply directly to the unrooted median tree case. The only obstacle is the precomputation algorithm, since it relies on the parent-child relationship defined by a root. However, we can easily adapt it to work with unrooted median trees by simply rooting them at *any* internal node. Note that rooting a tree at a node does not change its PD distance to any other tree.

5 Hybrid heuristic

As it was stated in the introduction, in a classical local search scenario there are two major phases. In the phase I a seed tree (starting supertree) is constructed incrementally. Typically, the process is initiated with some t' taxa, and an optimal tree over the chosen taxa is computed exactly. Here, t' is typically small, e.g., three. Next, on each iteration t new taxa are added to the partial supertree (here t is typically one) – an optimum among all possible ways to add t leaves to the tree is picked. The phase II is to run the actual local search starting with the seed tree obtained in phase I.

Clearly, the first phase could be rather slow, especially when it is costly to compute the distance measure for a supertree (as in our case). It can be done with a straightforward approach, where we try all possible positions to insert t new taxa in our partial tree, and directly calculate the distance after each possible insertion. Since there are $O(n^t)$ possible ways to insert t taxa, n taxa to insert overall and the distance calculation takes $O(kn^2)$ time – the overall complexity is $O(kn^{t+3})$, which is rather infeasible on large phylogenetic datasets even when $t = 1$.

Note, however, that we can improve the above complexity for $t = 1$ by employing the same SPR-neighborhood exploration idea as used in the core of our local search approach presented in section 4. More formally, let S be a partially constructed tree, and a be a taxon that we want to insert at a "best" position in S. Let now S' be a tree obtained by inserting a at a random position in S. Then, clearly, $SPR_{S'}(a)$ contains all the other possibilities of inserting a. Therefore, we can apply the efficient algorithm from section 4 to construct a starting tree for $t = 1$ in $O(kn^3)$ time.

Even though some other ideas could be suggested to accelerate the first phase, we want to emphasize that it is not necessary to separate the two phases in the first place. That is, the local search heuristic, which is the main optimization engine, could be applied on every step of construction of the starting tree (i.e., we add t new taxa randomly to a tree and then launch a local search procedure to improve it). It could be argued that SPR-based local search brings in more flexibility than simply trying to add new taxa to a tree with a fixed structure as it is done in the *two-phase* approach. For estimation of PD median trees we implemented this novel *hybrid* heuristic using the introduced here local search algorithm.

6 Experimental Evaluation

In this section we analyze scalability and accuracy of our novel path-difference median tree (PDM) algorithm when implemented in various heuristic settings, both on simulated and empirical data sets.

Experiments on simulated data clearly outline time efficiency and scalability of our algorithm (as well as the hybrid heuristic) and demonstrate its applicability to truly large-scale phylogenetic studies. Furthermore, our empirical study demonstrates the effectiveness of the developed hybrid heuristic as opposed to the traditional two-phase heuristic and other popular supertree methods.

6.1 Scalability analysis

Here we report the scalability analysis of phase II that is implementing our new PDM algorithm in comparison with implementing the previously best-known SPR-based local search algorithm for this phase [7].

6.1.1 Implementation and configuration

Both local search algorithms were implemented using Python. For the experimental evaluation the PyPy python interpreter was used under Windows 7 on an Intel Core i7 2.5GHz CPU.

To conduct our performance evaluation we designed several experiments on simulated data by generating multiple random tree-inputs with 10 trees each, but with a varying number of taxa (from 10 taxa up to 100 taxa with an increase of 10 taxa).

6.1.2 Experimental setting

Here we are analyzing the runtime performance of the local search algorithms in phase II only and in the hybrid heuristic.

Scalability of phase II. We compare the two algorithms by running them only in phase II on randomly generated seed trees. That way we can compare directly the performance of the two methods avoiding a bias introduced by a more advanced choice of a starting tree (seed tree) like in the standard heuristic or hybrid heurisitic. An additional advantage of such analysis is that it shows how fast we can expect a random supertree to converge to a local minimum in the SPR-graph.

Hybrid heuristic scalability. In this paper we put forward a *hybrid* local search approach. Thus we compare performance of the two algorithms as applied to a hybrid heuristic. In both cases we set $t = 10$ (meaning that we add 10 taxa to a candidate median tree per step).

6.1.3 Results and discussion

The results of both experiments are depicted in Figure 2. We observe that in both cases, as expected, our new PDM algorithm's runtime grows considerably slower comparing to the previously best known method. In addition, when we compare the mean runtime of a random restart launch and a hybrid heuristic launch, the later turns out to be noticeably faster (even though the hybrid approach includes a computationally-heavy step of a seed tree construction).

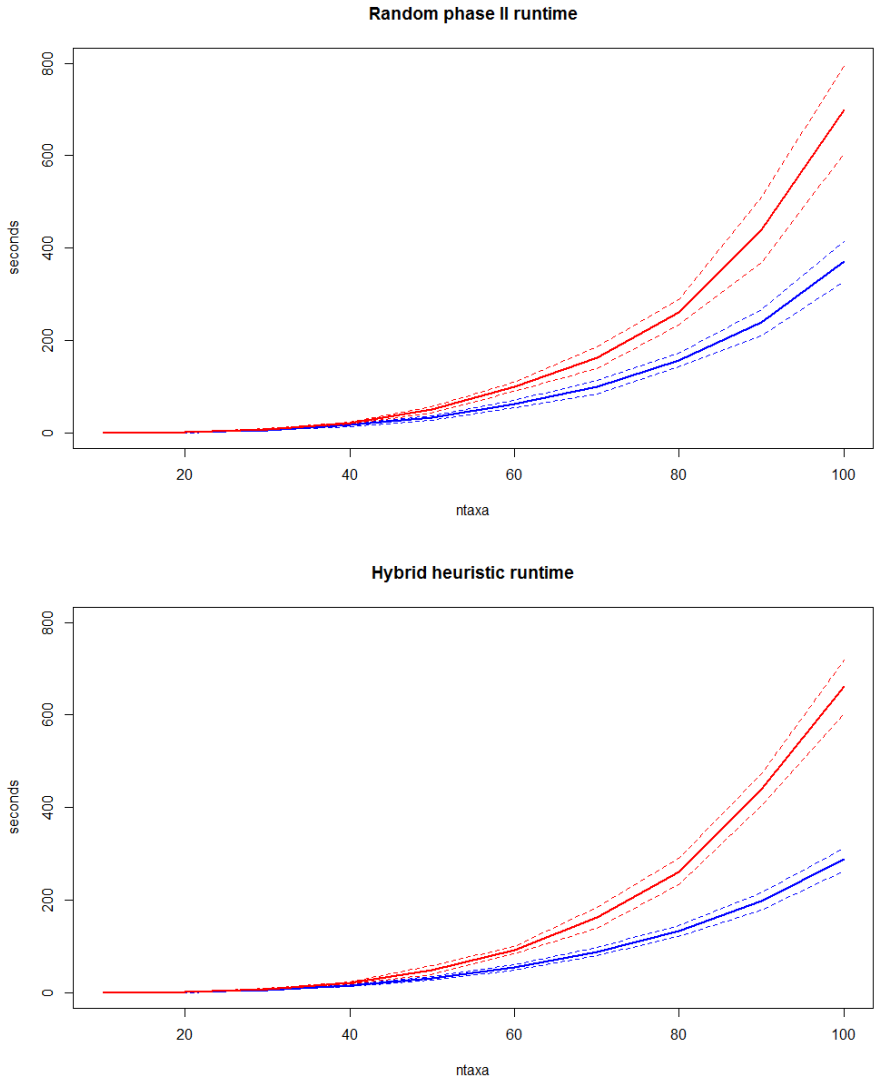


Fig. 2: Phase II and the hybrid heuristic were launched 20 times on each simulated input. The *solid* blue and red lines show the resulting *mean* runtime of the heuristics under our new $\Theta(kn^2)$ local search algorithm and the previously best-known $\Theta(kn^3)$ algorithm [7] correspondingly. The dashed lines depict the mean runtime $\pm$ the sample *standard deviation* over the 20 runs computed at each point.

6.2 Empirical evaluation

For an empirical study we adhere to a classical evaluation approach by comparing our path difference median tree heuristic against standard supertree methods with different objectives [25], [26].

6.2.1 Data sets

We processed two published baseline phylogenetic datasets, the Marsupials dataset [27] and the Cetartiodactyla dataset [28]. These datasets have been actively used for experimental supertree evaluations throughout the evolutionary community (see, for example, [26], [29], [30], [31]).

6.2.2 Experimental setting

Following the experimental settings presented in one of the recent supertree papers [29], we compare our PDM algorithm against the following supertree methods: the maximum representation with parsimony (MRP) heuristic [32], the modified min-cut (MMC) algorithm [33], and the triplet supertree heuristic [29].

MRP heuristics are addressing the NP-hard MRP problem [25], and are among the most popular supertree methods in evolutionary biology [6]. For our evaluation we use the MRP local search heuristic implemented in PAUP* [32] with Tree Bisection and Reconnection (TBR) branch swapping [29]. The *TBR edit operation* is an extension of the SPR operation, where the pruned subtree is allowed to be re-rooted before regrafting it. The MMC algorithm computes supertrees (that satisfy certain desirable properties) in polynomial time, which makes this method especially attractive for large-scale phylogenetic analysis [33]. The triplet supertree heuristic is a local search heuristic that is addressing the well-studied NP-hard triplet supertree problem [29]. We are using the triplet heuristic based on SPR and TBR local searches, called TH(SPR) and TH(TBR) respectively.

Finally, in order to justify the significance of our *hybrid*

TABLE 2: Summary of the experimental evaluation. The best scores under each objective are highlighted in bold.

Data set	Method	PD distance	Triplet-sim	MAST-sim	Pars. score
Marsuplial 158 input trees 272 taxa	MMC	16,670.45	51.73 %	53.4 %	3901
	MRP	5,694.59	98.29 %	**71.6** %	**2274**
	TH(SPR)	5,866.27	**98.99** %	70.0 %	2312
	TH(TBR)	5,888.22	**98.99** %	70.1 %	2317
	PDM(2P)	4,462.18	84.95 %	67.4 %	2912
	PDM(H)	**4,380.77**	85.24 %	67.0 %	2869
Cetartiodactyla 201 input trees 299 taxa	MMC	16,206.17	70.03 %	51.5 %	4929
	MRP	6,991.36	96.49 %	**65.2** %	**2603**
	TH(SPR)	7,630.03	**97.28** %	63.1 %	2754
	TH(TBR)	7,591.13	**97.28** %	63.0 %	2754
	PDM(2P)	5,742.37	83.97 %	59.8 %	3654
	PDM(H)	**5,639.24**	85.98 %	61.0 %	3394

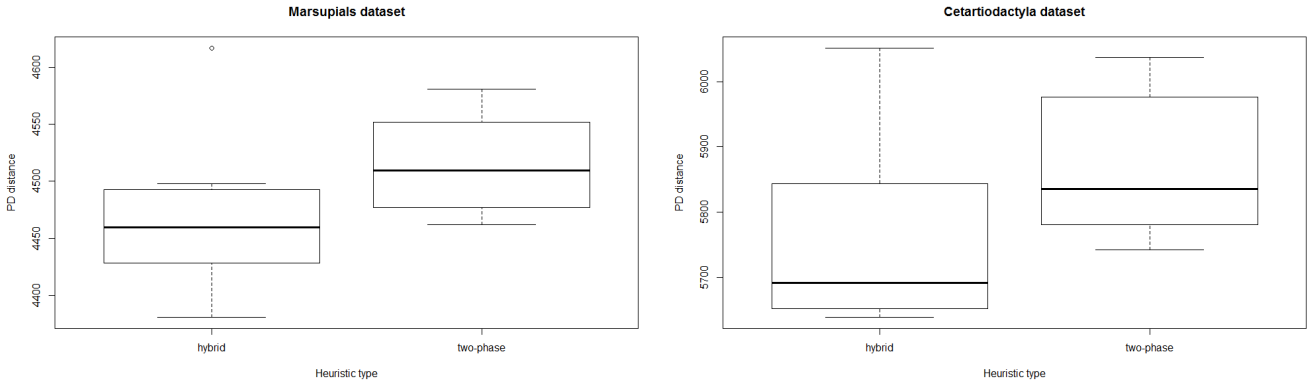


Fig. 3: Each box corresponds to distances of 10 median trees generated by either PDM(H) or PDM(2P) methods.

heuristic we evaluate our PDM algorithm as applied separately to the hybrid heuristic PDM(H) and the traditional two-phase heuristic PDM(2P).

For the experiments 10 supertrees were constructed by each method (except for MMC, since it is a *deterministic* algorithm and we only need a single tree produced by it) for each dataset. Then a best score under each objective among 10 trees was used and reported in the Table 2. Also note that for the PDM(H) heuristic we used a setting of $t = 10$ (i.e., adding 10 taxa per step to the a candidate median tree).

6.2.3 Results and Discussion

We first discuss the results from cross-validating the objective scores of the here used supertree methods. Then we compare the supertree estimates produced by these methods using standard tree distance measures. Finally, we contemplate the supertree objective scores with an estimated uniform distribution of the path distance.

Supertree method comparison. Table 2 summarizes the results that we obtained from the conducted experiments with our heuristics PDM(H) and PDM(2P) when compared to the published results for MMC, MRP, TH(SPR), and TH(TBR) [29].

As expected, all of the methods stand their ground. The MRP method proves to be most effective according to the parsimony objective. In addition, MRP supertrees show the best fit over the input data in terms of our computed MAST-similarity scores – which could be seen as an "independent" objective in our evaluation. At the same time, both triplet heuristics, TH(SPR) and TH(TBR), inferred the best supertree estimates under the triplet similarity objective. As for our methods – PDM(H) and PDM(2P) – they were able to infer the best supertree estimates with regards to the PD distance.

Furthermore, we observe that the hybrid approach (PDM(H)) outperformed the traditional approach (PDM(2P)) on both datasets. To confirm that the significance of this result we present box diagrams with distances for all generated PD median trees (see Figure 3)

From Figure 3 we see that a median PD distance for the PDM(H) method is notably lower than for the PDM(2P) method on both datasets.

Supertree comparison. From Table 2 we see how well supertrees computed by different methods fit the input datasets. However, it would be also valuable to see how these supertrees are different. This comparison can give us an idea on how diverse the methods and their corresponding objectives are.

To compare the computed supertrees, we choose a single best-fitting tree for each method (e.g., among the 10 computed supertrees for MRP we choose a one that minimizes the parsimony objective). Next we compare the chosen supertrees with the *Robinson-Foulds (RF)* metric and the triplet similarity measure. The results are presented in Tables 3,4, and 5.

Note that in Table 3 we normalize the RF scores by

TABLE 3: Normalized *Robinson-Foulds* scores between supertree estimates constructed via different phylogenetic methods.

		PDM(H)	PDM(2P)	MRP	TH(SPR)	TH(TBR)	MMC
Marsupials	PDM(H)	0	0.81	0.79	0.79	0.8	0.82
	PDM(2P)	0.81	0	0.8	0.79	0.8	0.81
	MRP	0.79	0.8	0	0.35	0.33	0.8
	TH(SPR)	0.79	0.79	0.35	0	0.24	0.79
	TH(TBR)	0.8	0.8	0.33	0.24	0	0.81
	MMC	0.82	0.81	0.8	0.79	0.81	0
Cetartiodactyla	PDM(H)	0	0.83	0.82	0.83	0.83	0.89
	PDM(2P)	0.83	0	0.82	0.84	0.85	0.9
	MRP	0.82	0.82	0	0.37	0.38	0.81
	TH(SPR)	0.83	0.84	0.37	0	0.14	0.82
	TH(TBR)	0.83	0.85	0.38	0.14	0	0.84
	MMC	0.89	0.9	0.81	0.82	0.84	0

TABLE 4: *Triplet similarity* scores between supertree estimates constructed via different phylogenetic methods. Maximum possible similarity score is 100

		PDM(H)	PDM(2P)	MRP	TH(SPR)	TH(TBR)	MMC
Marsupials	PDM(H)	0	73.23	76.94	80.77	80.77	49.81
	PDM(2P)	73.23	0	75.23	76.66	76.66	45.51
	MRP	76.94	75.23	0	90.39	90.39	50.79
	TH(SPR)	80.77	76.66	90.39	0	99.98	53.32
	TH(TBR)	80.77	76.66	90.4	99.98	0	53.32
	MMC	49.32	45.07	50.29	52.8	52.8	0
Cetartiodactyla	PDM(H)	0	78.89	86.68	85.78	85.79	69.01
	PDM(2P)	78.89	0	79.82	82.59	82.59	63.0
	MRP	86.68	79.82	0	91.12	91.12	74.61
	TH(SPR)	85.78	82.59	91.12	0	99.99	69.18
	TH(TBR)	85.79	82.59	91.12	99.99	0	69.18
	MMC	68.96	62.97	74.57	69.14	69.14	0

TABLE 5: *MAST similarity* scores between supertree estimates constructed via different phylogenetic methods. Maximum possible similarity score is 1

		PDM(H)	PDM(2P)	MRP	TH(SPR)	TH(TBR)	MMC
Marsupials	PDM(H)	0	0.4301	0.4154	0.4081	0.4044	0.1949
	PDM(2P)	0.4301	0	0.4007	0.4632	0.4632	0.1838
	MRP	0.4154	0.4007	0	0.6029	0.6103	0.2243
	TH(SPR)	0.4081	0.4632	0.6029	0	0.8272	0.2279
	TH(TBR)	0.4044	0.4632	0.6103	0.8272	0	0.2279
	MMC	0.1949	0.1838	0.2243	0.2279	0.2279	0
Cetartiodactyla	PDM(H)	0	0.3612	0.4181	0.3679	0.3712	0.1605
	PDM(2P)	0.3612	0	0.3579	0.3712	0.3679	0.1438
	MRP	0.4181	0.3579	0	0.5385	0.5518	0.2843
	TH(SPR)	0.3679	0.3712	0.5385	0	0.9097	0.2408
	TH(TBR)	0.3712	0.3679	0.5518	0.9097	0	0.2375
	MMC	0.1605	0.1438	0.2843	0.2408	0.2375	0

$2n - 4$, which is the maximal possible RF score for two *rooted* trees. The RF metric, while being the most applied tree comparison tool in computational biology, is widely criticized due to its unappealing distribution properties, where most of the trees have a very large RF score to a base tree [34]. This property, in particular, makes it hard to interpret and compare the resulting scores. Therefore, in addition to RF we use another widely applied comparison measurement for rooted trees – triplet similarity – which also has more attractive distribution properties. Finally, we use the maximum agreement subtree (MAST) similarity for its direct interpretability.

From all three tables we observe that MRP, TH(SPR) and TH(TBR) trees are closely related, while MMC and PDM trees appear to be rather different from them in terms of RF scores, triplet similarity, and MAST similarity. Moreover, we see that even PDM(2P) and PDM(H) trees are considerably different under the chosen objectives, while they fit comparably well the input data in terms of the PD distance. Finally, all three tables indicate that the MMC supertree is

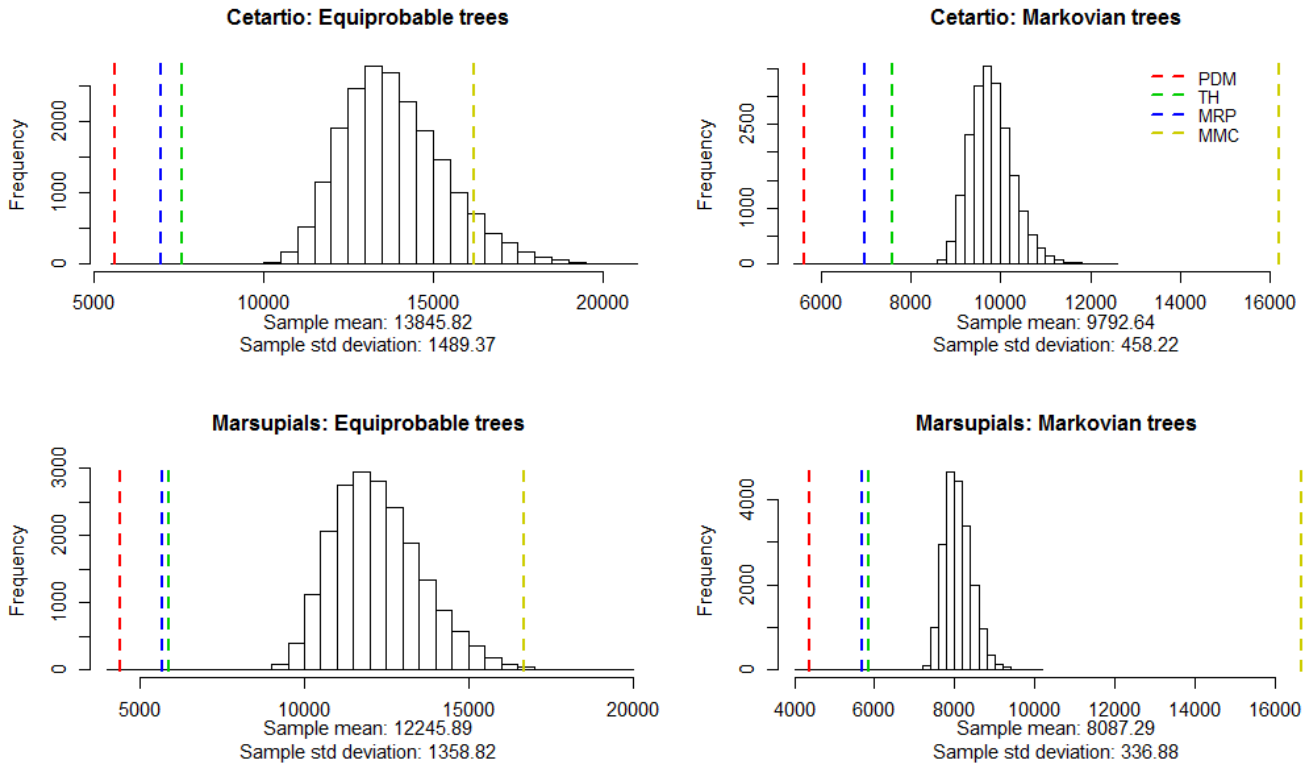


Fig. 4: Histograms of the PD distance based on the generated tree samples. All methods used for the evaluation are marked on each histogram with dotted lines.

least correlated with the supertree estimates resulting from the other methods.

Distribution. To further assess the results of our heuristics, we should contemplate them with the distribution of the path-difference distance for the two datasets. However, such distributions, even for a single input tree, remain unknown [12]. Thus, following the approach from Steel and Penny [9], we estimate PD distance distributions based on sample data. For each dataset we generated two collections with $20,000$ random supertrees using PAUP*. One collection was generated under the uniform binary tree distribution, and the other one was generated using the Markovian branching process [35]. Then, each collection was processed to obtain sample datasets with PD distance scores for every generated supertree. The obtained results are outlined in Figure 4.

The figure makes it clear that even though our heuristic was able to obtain the best results for the two datasets, MRP and Triplet heuristics also produce trees that are significantly better than any of the randomly generated trees under the PD objective. As for the MMC algorithm, it performs much worse in terms of PD distance than simply constructing Markovian Binary trees; and, what is more, generally worse than drawing random trees from the uniform distribution. In addition it is worth noting that Markovian binary trees appear to be biased towards the PD distance objective (i.e., the sample mean among Markovian trees is significantly lower than a sample mean for uniformly distributed trees on both datasets).

Figure 4 suggests that there exists a positive correlation between the Parsimony, Triplet-similarity, and PD distance supertree objectives. On the other hand, according to Table 2, better PD supertrees do not necessarily score well in terms of parsimony and triplet measures. Thus, our PD heuristic might produce structurally new phylogenetic trees that have not been analyzed previously (this is additionally confirmed by the Tables 3,4, and 5).

7 Conclusion

There is a rising interest in computation of median trees under one of the oldest and widely popular tree distance metrics – the path-difference distance. While the corresponding PD median tree problem is NP-hard, it was shown that it can be successfully approached by using an SPR based local search heuristic. In this work we presented a substantially improved local search algorithm comparing to the previously best known solution. In our study on simulated phylogenetic data we showed the advantage of our novel local search algorithm and demonstrated that it can be applicable to truly large-scale empirical phylogenetic datasets.

Further, the experiments on empirical data established the significance of the developed hybrid local search heuristic (that was enabled for large-scale analysis by our improved algorithm) and indicated that path-difference median trees can be expected to be structurally different from the de facto standard and widely applied MRP supertrees.

Currently, no mainstream supertree method can construct edge-weighted supertrees. However, there has been

an increased interest in such tools due to fast developing databases of time-annotated evolutionary trees (e.g., TimeTree [36]). The path-difference distance on the other hand, is naturally extendable to account for edge-weights in phylogenetic trees. Thus, the local search approach to the *weighted* path-difference median tree problem might be of great interest. This property makes the PD distance even more appealing as a median tree objective and suggests further investigation in its theoretical and algorithmic means.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments that helped to improve the quality of this work. This material is based upon work supported by the National Science Foundation under Grant No. 1617626.

REFERENCES

[1] S. Nik-Zainal, P. Van Loo, D. C. Wedge, L. B. Alexandrov, C. D. Greenman, K. W. Lau, K. Raine, D. Jones, J. Marshall, M. Ramakrishna, A. Shlien, S. L. Cooke, J. Hinton, A. Menzies, L. A. Stebbings, C. Leroy, M. Jia, R. Rance, L. J. Mudie, S. J. Gamble, P. J. Stephens, S. McLaren, P. S. Tarpey, E. Papaemmanuil, H. R. Davies, I. Varela, D. J. McBride, G. R. Bignell, K. Leung, A. P. Butler, J. W. Teague, S. Martin, G. Jönsson, O. Mariani, S. Boyault, P. Miron, A. Fatima, A. Langerød, S. A. J. R. Aparicio, A. Tutt, A. M. Sieuwerts, Å. Borg, G. Thomas, A. V. Salomon, A. L. Richardson, A.-L. Børresen-Dale, P. A. Futreal, M. R. Stratton, P. J. Campbell, and Breast Cancer Working Group of the International Cancer Genome Consortium, "The life history of 21 breast cancers," *Cell*, vol. 149, no. 5, pp. 994–1007, May 2012.

[2] R. A. Hufbauer, R. A. Marrs, A. K. Jackson, R. Sforza, H. P. Bais, J. M. Vivanco, and S. E. Carney, "Population structure, ploidy levels and allelopathy of Centaurea maculosa (spotted knapweed) and C. diffusa (diffuse knapweed) in North America and Eurasia," in *Proceedings of the XI International Symposium on Biological Control of Weeds, Canberra Australia*. Morgantown, WV.: USDA Forest Service. Forest Health Technology Enterprise Team, 2003, pp. 121–126.

[3] S. R. Harris, E. J. Cartwright, M. E. Török, M. T. Holden, N. M. Brown, A. L. Ogilvy-Stuart, M. J. Ellington, M. A. Quail, S. D. Bentley, J. Parkhill, and S. J. Peacock, "Whole-genome sequencing for analysis of an outbreak of meticillin-resistant staphylococcus aureus: a descriptive study," *Lancet Infect Dis*, vol. 13, no. 2, pp. 130–6, 2013.

[4] A. P. Jackson, "A reconciliation analysis of host switching in plant-fungal symbioses," *Evolution*, vol. 58, no. 9, pp. 1909–23, 2004.

[5] R. D. Page and E. Holmes, *Molecular evolution: a phylogenetic approach*. Blackwell Science, 1998.

[6] O. R. Bininda-Emonds, Ed., *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, ser. Computational Biology. Springer Verlag, 2004, vol. 4.

[7] A. Markin and O. Eulenstein, "Path-difference median trees," in *Bioinformatics Research and Applications: 12th International Symposium, ISBRA 2016, Minsk, Belarus, June 5-8, 2016, Proceedings*, A. Bourgeois, P. Skums, X. Wan, and A. Zelikovsky, Eds. Cham: Springer International Publishing, 2016, pp. 211–223.

[8] J. Farris, "A successive approximations approach to character weighting." *Systematic Zoology*, vol. 18, pp. 374–385, 1969.

[9] M. A. Steel and D. Penny, "Distributions of tree comparison metrics - some new results," *Systematic Biology*, vol. 42, no. 2, pp. 126–141, 1993.

[10] J. Bluis and D. Shin, "Nodal distance algorithm: Calculating a phylogenetic tree comparison metric," in *3rd IEEE International Symposium on BioInformatics and BioEngineering (BIBE 2003), 10-12 March 2003, Bethesda, MD, USA*. IEEE Computer Society, 2003, pp. 87–94.

[11] P. Puigbò, S. Garcia-Vallvé, and J. O. McInerney, "TOPD/FMTS: a new software to compare phylogenetic trees," *Bioinformatics*, vol. 23, no. 12, pp. 1556–1558, 2007.

[12] A. Mir and F. Rosselló, "The mean value of the squared path-difference distance for rooted phylogenetic trees," *CoRR*, vol. abs/0906.2470, 2009.

[13] W. Williams and H. Clifford, "On the Comparison of Two Classifications of the Same Set of Elements," *Taxon*, vol. 20, no. 4, pp. 519–522, 1971.

[14] J. B. Phipps, "Dendogram topology," *Systematic Zoology*, vol. 20, pp. 306–308, 1971.

[15] W. P. Maddison and L. L. Knowles, "Inferring phylogeny despite incomplete lineage sorting," *Syst Biol*, vol. 55, no. 1, pp. 21–30, 2006.

[16] C. Than and L. Nakhleh, "Species tree inference by minimizing deep coalescences," *PLoS Comput Biol*, vol. 5, no. 9, p. e1000501, 2009.

[17] M. S. Bansal, J. G. Burleigh, and O. Eulenstein, "Efficient genome-scale phylogenetic analysis under the duplication-loss and deep coalescence cost models," *BMC Bioinformatics*, vol. 11 Suppl 1, p. S42, 2010.

[18] R. Chaudhary, M. S. Bansal, A. Wehe, D. Fernández-Baca, and O. Eulenstein, "iGTP: a software package for large-scale gene tree parsimony analysis," *BMC Bioinformatics*, vol. 11, p. 574, 2010.

[19] H. T. Lin, J. G. Burleigh, and O. Eulenstein, "Consensus properties for the deep coalescence problem and their application for scalable tree search," *BMC Bioinformatics*, vol. 13 Suppl 10, p. S12, 2012.

[20] C. Semple and M. A. Steel, *Phylogenetics*. Oxford: University Press, 2003.

[21] R. Chaudhari, G. J. Burleigh, and O. Eulenstein, "Efficient Algorithms for Rapid Error Correction for Gene Tree Reconciliation using Gene Duplications, Gene Duplication and Loss, and Deep Coalescence," *BMC Bioinformatics*, vol. 13 Suppl 10, p. S11, 2012.

[22] J. A. Cotton and M. Wilkinson, "Majority-rule supertrees," *Syst Biol*, vol. 56, no. 3, pp. 445–452, 2007.

[23] D. Bryant, "Hunting for trees in binary character sets: efficient algorithms for extraction, enumeration, and optimization." *J Comput Biol*, vol. 3, no. 2, pp. 275–288, 1996.

[24] M. A. Steel, "The complexity of reconstructing trees from qualitative characters and subtrees," *Journal of Classification*, vol. 9, pp. 91–116, 1992.

[25] S. Moran, S. Rao, and S. Snir, "Using semi-definite programming to enhance supertree resolvability," in *Proceedings of the 5th International Conference on Algorithms in Bioinformatics*, ser. WABI'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 89–103.

[26] M. S. Bansal, J. G. Burleigh, O. Eulenstein, and D. Fernández-Baca, "Robinson-foulds supertrees," *Algorithms for Molecular Biology*, vol. 5, no. 1, pp. 1–12, 2010.

[27] M. Cardillo, O. R. P. Bininda-Emonds, E. Boakes, and A. Purvis, "A species-level phylogenetic supertree of marsupials," *Journal of Zoology*, vol. 264, pp. 11–31, 2004.

[28] S. A. Price, O. R. P. Bininda-Emonds, and J. L. Gittleman, "A complete phylogeny of the whales, dolphins and even-toed hoofed mammals (cetartiodactyla)," *Biological Reviews*, vol. 80, no. 3, pp. 445–473, 2005.

[29] H. T. Lin, J. G. Burleigh, and O. Eulenstein, "Triplet supertree heuristics for the tree of life," *BMC Bioinformatics*, vol. 10, no. Suppl 1, 2009.

[30] D. Chen, O. Eulenstein, D. Fernández-Baca, and J. Burleigh, "Improved heuristics for minimum-flip supertree construction," *Evolutionary Bioinformatics*, vol. 2, 2006.

[31] S. Snir and S. Rao, "Quartets maxcut: A divide and conquer quartets algorithm," *IEEE/ACM TCBB*, vol. 7, no. 4, pp. 704–718, 2010.

[32] D. L. Swofford, "PAUP*. Phylogenetic analysis using parsimony (*and other methods). Version 4. Sinauer Associates, Sunderland, Massachusetts." 2002.

[33] R. D. M. Page, "Modified mincut supertrees," in *Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, ser. WABI '02. London, UK, UK: Springer-Verlag, 2002, pp. 537–552.

[34] D. Bryant and M. A. Steel, "Computing the Distribution of a Tree Metric," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 6, no. 3, pp. 420–426, 2009. [Online]. Available: http://doi.acm.org/10.1145/1577987.1577993

[35] N. G. Bean, N. Kontoleon, and P. G. Taylor, "Markovian trees: properties and algorithms," *Annals of Operations Research*, vol. 160, no. 1, pp. 31–50, 2007.

[36] A. D. Leaché, "The timetree of life. S. Blair Hedges and Sudhir Kumar, editors." *Integrative and Comparative Biology*, vol. 50, no. 1, pp. 141–142, 2010.

Alexey Markin received a B.S. degree in Computer Science from Higher School of Economics (Russia) in 2015. Since then he is a Ph.D. student of Computer Science at Iowa State University, where he works with Prof. Eulenstein on computational problems in biology with focus on evolutionary tree inference. His research interests include graph theory, statistics, and phylogenetics.

Oliver Eulenstein is a professor of computer science at Iowa State University. He earned his doctoral degree at the University of Bonn (Germany) in 1998, and held a postdoctoral position at the University of California Davis before joining the department of Computer Science at Iowa State University in 2000. His research interest is in Combinatorial Optimization, with special emphasis on Computational Biology and Bioinformatics.