Journal of Mathematical Biology (2021) 82:8
https://doi.org/10.1007/s00285-021-01567-5

Mathematical Biology

Computing nearest neighbour interchange distances between ranked phylogenetic trees

Lena Collienne[1] · Alex Gavryushkin[1]

Received: 20 August 2020 / Revised: 20 October 2020 / Accepted: 7 January 2021

Abstract

Many popular algorithms for searching the space of leaf-labelled (phylogenetic) trees are based on tree rearrangement operations. Under any such operation, the problem is reduced to searching a graph where vertices are trees and (undirected) edges are given by pairs of trees connected by one rearrangement operation (sometimes called a move). Most popular are the classical nearest neighbour interchange, subtree prune and regraft, and tree bisection and reconnection moves. The problem of computing distances, however, is **NP**-hard in each of these graphs, making tree inference and comparison algorithms challenging to design in practice. Although ranked phylogenetic trees are one of the central objects of interest in applications such as cancer research, immunology, and epidemiology, the computational complexity of the shortest path problem for these trees remained unsolved for decades. In this paper, we settle this problem for the ranked nearest neighbour interchange operation by establishing that the complexity depends on the weight difference between the two types of tree rearrangements (rank moves and edge moves), and varies from quadratic, which is the lowest possible complexity for this problem, to **NP**-hard, which is the highest. In particular, our result provides the first example of a phylogenetic tree rearrangement operation for which shortest paths, and hence the distance, can be computed efficiently. Specifically, our algorithm scales to trees with tens of thousands of leaves (and likely hundreds of thousands if implemented efficiently).

We thank Alexei Drummond, David Bryant, and Kieran Elmes for useful discussions about the weight difference between RNNI moves, complexity, and applied aspects of our results. Their comments improved our paper. We acknowledge support from the Royal Society Te Apārangi through a Rutherford Discovery Fellowship (RDF-UOO1702). This work was partially supported by Ministry of Business, Innovation, and Employment of New Zealand through an Endeavour Smart Ideas Grant (UOOX1912) and a Data Science Programmes Grant (UOAX1932).

✉ Alex Gavryushkin
alex@biods.org

Lena Collienne
lena.collienne@postgrad.otago.ac.nz

[1] Department of Computer Science, University of Otago, Dunedin, New Zealand

Published online: 25 January 2021

Mathematics Subject Classification 68Q25 · 92B05

One of the major problems in computational biology is the reconstruction of evolutionary histories, also known as phylogenetic trees, from sequence data such as RNA, DNA, or protein sequences. Of particular interest in various applications is the order of internal nodes in these trees, as these nodes represent evolutionary events and their ranking models the order in which these events happened in time. For example in species evolution, where internal nodes of trees correspond to speciation events, the ranking of these nodes represents the order of divergence events in time. Fossils can be used to rank and time divergence events in phylogenetic trees (Gavryushkina et al. 2014). Other research fields where ranked trees play an important role are viral epidemiology, where ranking gives the order of transmission events (Ypma et al. 2013), and language evolution (Bouckaert et al. 2018; Gray et al. 2009), where phylogenetic trees reveal how and when human populations expanded across different continents. Recently, phylogenetic trees have become a popular tool to study cancer evolution (Singer et al. 2018; Alves et al. 2019). In cancer phylogenies internal nodes can refer to emergence of metastatic clones and their ranking shows in which order metastases had been seeded in time (Lote et al. 2017).

Most commonly trees are inferred from sequences via maximum likelihood (Stamatakis 2006; Guindon et al. 2010), MCMC (Ronquist and Huelsenbeck 2003; Suchard et al. 2018; Bouckaert et al. 2019), distance-, or parsimony-based approaches (Tamura et al. 2011). A similarity measure between trees is required for the development of algorithms implementing these methods and evaluating the accuracy of reconstructed trees. Furthermore, summary or consensus tree methods (McMorris and Steel 1994; Bansal et al. 2010; Whidden et al. 2014) often rely on a tree metric. Most of the currently used distance measures for trees, however, do not take the order of divergence events into account—only the tree topology. Moreover, popular tree distances are either hard to compute or lack biological interpretability (Whidden and Matsen 2018).

Most tree inference methods rely on various tree rearrangement operations (Semple and Steel 2003), the most popular of which are nearest neighbour interchange (NNI), subtree prune and regraft (SPR), and tree bisection and reconnection (TBR). Under any such operation, the tree inference problem can be formulated as a graph search, where vertices are trees and edges are given by tree rearrangement operations. For search algorithms to be efficient, it is important to understand the geometry of these graphs. For example, basic geometric properties of the NNI graph have been successfully leveraged to speed up the maximum likelihood method (Nguyen et al. 2015). The most basic geometric characteristic that frequently arises in applications is the minimum number of rearrangements necessary to transform one tree into another (Semple and Steel 2003). The problem then amounts to computing the length of a shortest path between trees in the NNI, SPR, or TBR graph. This can also be seen as computing the distance between trees in the corresponding metric space.

Classical results in mathematical phylogenetics imply that these distances are **NP**-hard to compute for all three rearrangement operations NNI, SPR, and TBR (DasGupta et al. 2000; Bordewich and Semple 2005; Hickey et al. 2008; Allen and Steel 2001). Intuitively, the difference between them is how much change can be done to a tree

by a single operation, with NNI being the most local type of rearrangement and TBR the most global one. Remarkably, it took over 25 years and a number of published erroneous attempts, as discussed in detail by DasGupta et al. (2000), to prove that computing distances is **NP**-hard in NNI (DasGupta et al. 2000). Similarly, incorrect proofs for SPR have been discussed in the literature (Hein et al. 1996; Allen and Steel 2001), before Bordewich and Semple (2005) proved the **NP**-hardness result for rooted trees and Hickey et al. (2008) utilised this proof to establish the result for unrooted trees. To facilitate practical applications, fixed parameter tractable algorithms (Downey and Fellows 2013) for computing the SPR distance have been developed over the years (Whidden et al 2010; Bordewich and Semple 2005; Whidden and Matsen 2018). Computing the NNI distance is also known to be fixed parameter tractable (DasGupta et al. 1999). Although important, these algorithms remain impractical for large distances and are only applied to trees with a moderate number of leaves or those with small distances (Whidden and Matsen 2018).

Another popular tree distance measure that does not rely on a tree rearrangement method is the Robinson–Foulds distance (Robinson and Foulds 1981). In contrast to the tree rearrangement-based distances mentioned above, this distance can be computed efficiently. A downside of this approach however is a lack of biological interpretability. The Robinson–Foulds distance is not motivated by a biological process, unlike for example SPR, where the tree rearrangement operation can be used to model hybridisation and other horizontal events. This pattern is quite common—tree distance measures that are easy to compute lack biological interpretability, while those that are biologically meaningful are often hard to compute (Whidden and Matsen 2018).

In this paper, we consider a generalisation of the NNI operation to ranked trees introduced by Gavryushkin et al. (2018), which is called RNNI (for Ranked Nearest Neighbour Interchange). We show that the shortest path problem in RNNI is computable in $\mathcal{O}(n^2)$, where n is the number of tree leaves. This makes RNNI the first tree rearrangement operation under which shortest paths and distances between trees are polynomial-time computable. Our proof of this result (Theorem 1) is constructive—we provide an algorithm called FindPath that computes shortest paths in the RNNI graph in $\mathcal{O}(n^2)$ time. Our algorithm is optimal as shortest paths often have length quadratic in the number of leaves n. The algorithm is practical as it takes seconds on a laptop to compute the distance between trees with thousands of leaves, while in the closely related NNI graph the tractable number of leaves is well below twenty (Li et al. 1996; Whidden and Matsen 2017). Furthermore, FindPath reveals the following property of the RNNI graph, which is desirable for tree distances from a biological point of view. If two trees share some information, more specifically a cluster, there is a shortest paths in RNNI that preserves this information (the cluster). In other words, shortest paths in RNNI maintain clusters, an important property that is not true in NNI (Li et al. 1996). This implies in particular that trees that share an evolutionary hypothesis in form of a common subtree are closer to each other than they are to a tree not sharing a subtree with them. For a cancer phylogeny this can be interpreted as two trees supporting the emergence of one particular metastatic clone are closer to each other than a phylogeny that does not support this hypothesis.

Because NNI can be seen as a special case of RNNI, we investigate whether there exists a threshold at which the complexity of the shortest path problem shifts from

NP-hard to polynomial. Specifically, we introduce an edge weight parameter ρ in the RNNI graph and consider a parametrised graph RNNI(ρ). More precisely, the RNNI operations that change the ranking, but not the tree topology, weigh ρ, while moves that change the topology weigh one. We show that the shortest path problem is **NP**-hard in RNNI(0) and quadratic in RNNI(1), so the complexity changes with ρ. We hence propose to characterise the complexity classes of the problem RNNI(ρ) for values of $\rho \geq 0$.

The biological interpretation of this characterisation problem is as follows. In many large-scale applications two or more different methods are used to reconstruct an evolutionary process—one to model and reconstruct the branching process and another one to time or rank the evolutionary events (Lote et al. 2017). Often this results in different support probabilities for the inferred tree topologies and for the ranking of events. A comparison method for trees inferred this way has to have different penalties for conflicts in the tree structure and the ranking. This difference can be quantitatively modelled using our ρ parameter. For example, if the tree topology estimate is more certain than the ranking, ρ should be chosen to be less than one. An efficient algorithm to compare trees for such values of ρ is hence desirable.

1 Definitions and background results

Unless stated otherwise, by a *tree* in this paper we mean a *ranked phylogenetic tree*, which is a binary tree where leaves are uniquely labelled by elements of the set $\{a_1, \ldots, a_n\}$ for a fixed integer n, and all internal (non-leaf) nodes are uniquely *ranked* by elements of the set $\{1, \ldots, n-1\}$ so that each child has a strictly smaller rank than its parent. All leaves are assumed to have rank 0 but we only refer to the ranks of internal nodes throughout. In total there are $\frac{(n-1)!n!}{2^{n-1}}$ such trees on n leaves (Gavryushkin et al. 2018). Two trees are considered to be identical if there exists an isomorphism between them which preserves edges, leaf labels, and node rankings. For example, trees in Fig. 1 are all different.

Because internal nodes of a tree T are ranked uniquely, we can address **the** node of rank $t \in \{1, \ldots, n-1\}$, and we write $(T)_t$ to denote this node. An *interval* $[(T)_t, (T)_{t+1}]$ is defined by two nodes of consecutive ranks. A *cluster* $C \subseteq \{a_1, \ldots, a_n\}$ in a tree T is a subset of leaves that contains all leaves descending from one internal node of T. We then say that this internal node *induces* the cluster C, and that the subtree rooted at this node is *induced* by C. Trees can uniquely be specified using the *cluster representation*, that is a list of all clusters induced by internal nodes of that tree ordered according to the ranks of internal nodes. For example, the cluster representation of tree T in Fig. 1 is $[\{a_1, a_2\}, \{a_1, a_2, a_3\}, \{a_4, a_5\}, \{a_1, a_2, a_3, a_4, a_5\}]$. For a set $S \subseteq \{a_1, \ldots, a_n\}$ and tree T we denote the *most recent common ancestor* of S in T, that is the node of the lowest rank in T that induces a cluster containing all elements of S, by $(S)_T$. Note that $(C)_T = (T)_t$ if the cluster C is induced by the node of rank t in T.

Our main object of study is the following class of graphs RNNI(ρ) indexed by a real-valued parameter $\rho \geq 0$. Vertices of the RNNI(ρ) graph are trees as defined above. Two trees are connected by an edge (also called an RNNI *move*) if one results

from the other by performing one of the following two types of tree rearrangement operation (see Fig. 1):

(i) A *rank move* on a tree T exchanges the ranks of two internal nodes $(T)_t$ and $(T)_{t+1}$ with consecutive ranks, provided the two nodes are not connected by an edge in T.

(ii) Trees T and R are connected by an NNI *move* if there are edges e in T and f in R both connecting nodes of consecutive ranks in the corresponding trees, such that the (non-binary) trees obtained by shrinking e and f into internal nodes are identical.

The parameter $\rho \geq 0$ is the weight of the rank move operation, an NNI move weighs 1.

The *weight of a path* in RNNI(ρ) is the sum of the weights of all moves along the path. The *distance* between two trees in RNNI(ρ) is the weight of a path with the minimal weight, which we will call a *shortest path*. When $\rho = 1$ we assume that the graph is unweighted.

We consider the following class of problems parametrised by a real number $\rho \geq 0$.

RNNI(ρ)-SP INSTANCE: A pair of trees T and R on n leaves FIND: A path of minimal weight between T and R in RNNI(ρ)

Since RNNI(ρ) is a connected graph, there always exists a solution to RNNI(ρ)-SP. Furthermore, the size of every solution to an instance of RNNI(ρ)-SP is bounded by a polynomial in n, despite the search space being super-exponential. This is because the diameter of the RNNI(1) graph is bounded from above (Gavryushkin et al. 2018) by $n^2 - 3n - 5/8$.

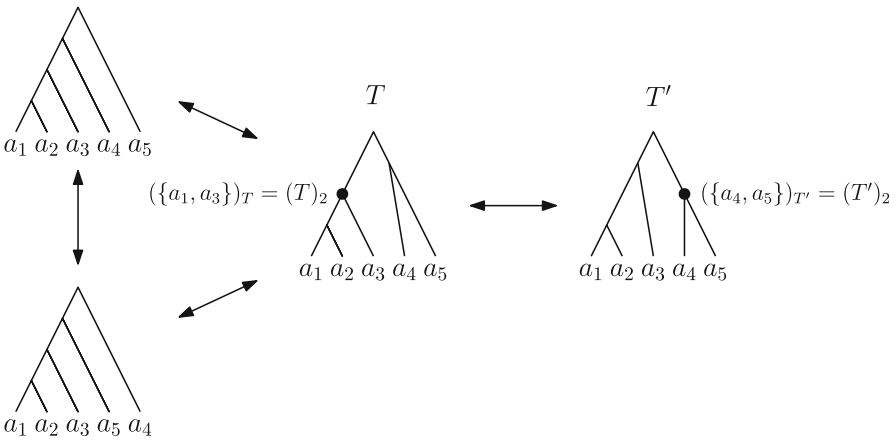


Fig. 1 Trees in the RNNI graph with three NNI moves on the left and a rank move on the right

Our main goal is to prove that RNNI(1)-SP can be solved in polynomial time. We will see later in the paper that it follows from a classical result (DasGupta et al. 2000) that RNNI(0)-SP is **NP**-hard. To be consistent with notations used in the literature (Gavryushkin et al. 2018), we will denote the graph RNNI(1) by RNNI.

2 FindPath algorithm

In this section we introduce an algorithm called FindPath that computes paths between trees and is quadratic in the number of leaves.

An input of the FindPath algorithm is two trees T and R in their cluster representation. We denote the representation of R by $[C_1, \ldots, C_{n-1}]$. The algorithm considers the clusters $C_1, \ldots, C_{n-2}$ iteratively in their order and produces a sequence p of trees which becomes a shortest path from T to R after the algorithm terminates. During each iteration $k = 1, \ldots, n-2$ new trees are added to p if necessary, and we will refer to the last added tree as T_1. In iteration k, the rank of $(C_k)_{T_1}$ is decreased by RNNI moves until C_k is induced by the node of rank k in T_1. In Proposition 1 we show that FindPath is a deterministic algorithm with running time quadratic in the number of leaves n. In particular, there always exists a unique move that decreases the rank of $(C_k)_{T_1}$ as described above.

Note that if two trees share a cluster, every tree on the path computed by FindPath contains this cluster as well. An implementation of this algorithm is available on GitHub (Collienne et al. 2019). Note that the version of FindPath implemented in (Collienne et al. 2019) outputs a shortest path as a list of trees. The algorithm that outputs the length of a shortest path can be implemented so that the wall clock running time on a generic laptop is under 30 s for trees with tens of thousands of leaves.

Algorithm 1 FindPath(T, R)

```
1: T_1 := T, p := [T_1], [C_1, ..., C_{n-1}] := R
2: for k = 1, ..., n − 2 do
3:    while rank((C_k)_{T_1}) > k do
4:       if (C_k)_{T_1} and node u with rank one less than (C_k)_{T_1} in T_1 are connected by an edge then
5:          T_2 is T_1 with the rank of (C_k)_{T_1} decreased by an NNI move
6:       else
7:          T_2 is T_1 with ranks of u and (C_k)_{T_1} swapped
8:       T_1 = T_2
9:       p = p + T_1
10: return p
```

Proposition 1 FindPath *is a correct deterministic algorithm that runs in* $\mathcal{O}(n^2)$ *time.*

Proof To show that FindPath is a deterministic algorithm (see the pseudocode above), we have to prove that tree T_2 constructed in the **while** loop (line 3) of the algorithm always exists and is uniquely defined. If T_2 is obtained in line 7 from T_1 by a rank move, the tree exists and is unique because there always exists exactly one rank move

on any particular interval that is not an edge. It remains to show that an NNI move that decreases the rank of $(C_k)_{T_1}$ always exists and is unique. To prove this we consider cases $k = 1$ and $k > 1$ separately.

Case $k = 1$. In this case C_k consists of two leaves $\{x, y\}$. Since we assumed that the **while** condition is satisfied, the node $v = (\{x, y\})_{T_1}$ has rank $r > 1$. Consider the node u with rank $r - 1$ in T_1. Assume without loss of generality that x is in the cluster induced by u, so y has to be outside this cluster. Consider the following three disjoint subtrees of T_1: the subtree T_{11} induced by a child of u and containing x, the subtree T_{12} induced by the other child of u, the subtree T_{13} induced by a child of v and containing y. Now observe that out of two NNI moves possible on the edge $[u, v]$ in T_1, only the one that swaps T_{12} and T_{13} does decrease the rank of the most recent common ancestor of $\{x, y\}$. Hence T_2 exists and is unique in this case.

Case $k > 1$. In this case $C_k = C_i \cup C_j$ for $i, j < k$. In this case the subtree of T_1 induced by $(C_i)_{T_1}$ is identical to the subtree of R induced by $(C_i)_R$, and the same is true for $(C_j)_{T_1}$ and $(C_j)_R$. Hence, we can reduce this case to $k = 1$ by suppressing C_i and C_j in both T_1 and R to new leaves c_i and c_j (of rank zero) respectively. As in Case $k = 1$, exactly one of two possible NNI moves deceases the rank of the most recent common ancestor of c_i, c_j in T_1, so the same is true for the most recent common ancestor $(C_k)_{T_1}$, and T_2 is unambiguously defined.

Thus, FindPath is a deterministic algorithm.

To prove correctness, note that the algorithm starts by adding T to the output path, and every new tree added to the output path is an RNNI neighbour of the previously added one (see line 5 and 7). To see that the output path terminates in R, observe that after k iteration of the **for** loop (line 2) of the algorithm, the first k clusters of T_1 and R must coincide, and so after $n - 2$ iterations a path between T and R is constructed.

The worst-case time complexity of FindPath is quadratic in the number of leaves, as there can be at most $n - 2$ executions of the **for** loop (line 2) and in every iteration of the **for** loop at most $n - 2$ **while** loops (line 3) are executed. Here and throughout the paper we assume that the output of FindPath is encoded by a list of RNNI moves rather than an actual list of trees. This is because writing out a tree on n leaves takes time linear in n and the complexity of FindPath becomes cubic. □

3 FindPath computes shortest paths in optimal time

In this section we prove the main result of this paper, that RNNI(1)-SP is polynomial. Specifically we prove that paths returned by FindPath are always shortest. We also show that FindPath is an optimal algorithm, that is, no sub-quadratic algorithm can solve RNNI(1)-SP.

The main ingredient of our proof is to show that a local property (see (1) in the proof) of the FindPath algorithm is enough to establish that the output paths are shortest. The property can intuitively be understood as FindPath always choosing

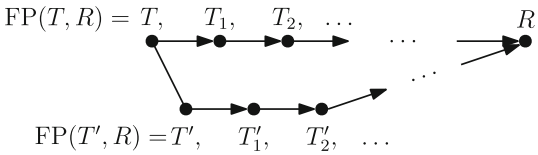


Fig. 2 Trees T, T', and R as in inequality (1). Paths $\mathrm{FP}(T, R) = [T, T_1, T_2, \ldots, R]$ and $\mathrm{FP}(T', R) = [T', T'_1, T'_2, \ldots, R]$ are indicated by arrows

the best tree possible to go to. Importantly, this result can be used for an arbitrary vertex proposal algorithm in an arbitrary graph to establish that the algorithm always follows a shortest path between vertices in the graph, hence our proof technique is of general interest.

Theorem 1 *The worst-case time complexity of the shortest path problem in the* RNNI *graph on trees with n leaves is* $\mathcal{O}(n^2)$. *Hence* RNNI(1)-SP *is polynomial time solvable.*

Proof We prove this theorem by showing that for every pair of trees T and R, the path computed by the FindPath algorithm is a shortest RNNI path. We denote this path by $\mathrm{FP}(T, R)$ and its length by $|\mathrm{FP}(T, R)|$. By $d(T, R)$ we denote the length of a shortest path between T and R, that is, the RNNI distance between trees. We hence want to show that $|\mathrm{FP}(T, R)| = d(T, R)$ for all trees.

Assume to the contrary that T and R are two trees with a minimum distance $d(T, R)$ such that $d(T, R) \neq |\mathrm{FP}(T, R)|$, that is, $d(T, R) < |\mathrm{FP}(T, R)|$. Let T' be the first tree on a shortest RNNI path from T to R. Then $d(T', R) = d(T, R) - 1$, implying that the distance between T' and R is strictly smaller than that between T and R. This implies that $|\mathrm{FP}(T', R)| = d(T', R) = d(T, R) - 1 < |\mathrm{FP}(T, R)| - 1$ and hence, $|\mathrm{FP}(T', R)| < |\mathrm{FP}(T, R)| - 1$. We finish the proof by showing that no trees satisfy this inequality.

Specifically, we will show that

$$\begin{gathered}\text{for all trees } T, R, \text{ and } T' \text{ such that } T' \text{ is one RNNI move away from } T,\\ |\mathrm{FP}(T', R)| \geq |\mathrm{FP}(T, R)| - 1\end{gathered} \tag{1}$$

We will use Fig. 2 to demonstrate our argument.

Assume to the contrary that T and R are trees for which there exists T' violating inequality (1). Out of all such pairs T, R choose one with the minimal $|\mathrm{FP}(T, R)|$. Denote $\mathrm{FP}(T, R) = [T, T_1, T_2, \ldots, R]$ and $\mathrm{FP}(T', R) = [T', T'_1, T'_2, \ldots, R]$, and let $[(T)_t, (T)_{t+1}]$ be the interval in T on which the RNNI move connecting T and T' is performed. Let C_k be the cluster of R such that the node $(C_k)_T$ is moved down by the first move on $\mathrm{FP}(T, R)$. If the rank of $(C_k)_T$ is not in $\{t, t+1\}$ then $(C_k)_T$ and $(C_k)_{T'}$ induce the same cluster, so FindPath would make the same rearrangement in both trees T and T' in the first move along $\mathrm{FP}(T, R)$ and $\mathrm{FP}(T', R)$ resulting in trees T_1 and T'_1 which are RNNI neighbours, as in Fig. 2. In this case, paths $\mathrm{FP}(T_1, R)$ and $\mathrm{FP}(T'_1, R)$ violate inequality (1) but $\mathrm{FP}(T_1, R)$ is strictly shorter than $\mathrm{FP}(T, R)$, contradicting our minimality assumption. Hence, the first move on $\mathrm{FP}(T, R)$ has to involve an interval incident to at least one of the nodes $(T)_t, (T)_{t+1}$.

Moreover, because C_k is the first cluster satisfying the **while** condition of FindPath applied to T and R, all clusters C_j with $j < k$ have to be present in T. And since the

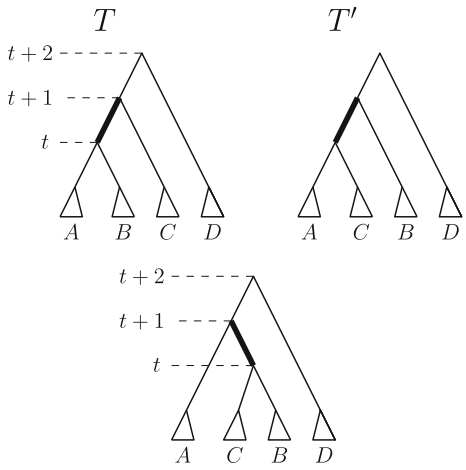


Fig. 3 NNI move between T and T' on the edge $[(T)_t, (T)_{t+1}]$ indicated in bold, and the third RNNI neighbour resulting from a move on this edge

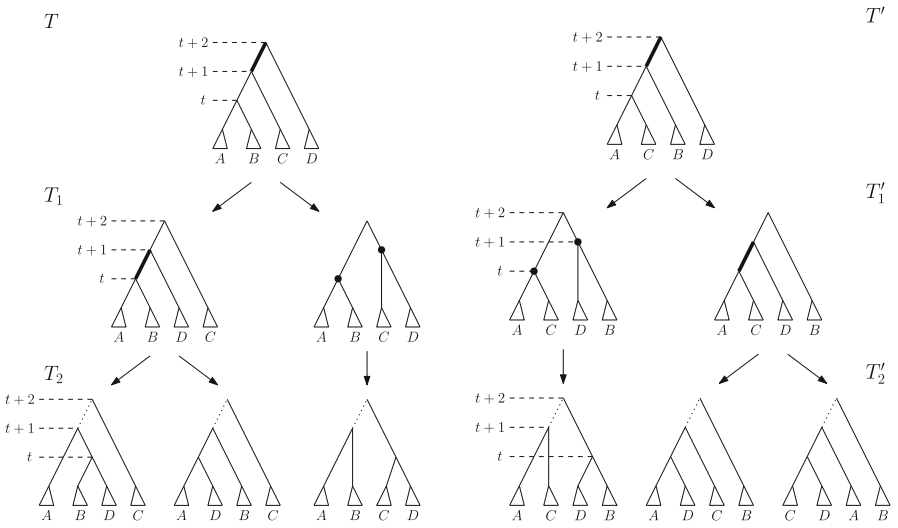
first move on FP(T, R), which decreases the rank of $(C_k)_T$, involves nodes with ranks not higher than $t+2$, the most recent common ancestor of C_k has rank not higher than $t+1$ after this move. Hence $k \leq t+1$. Furthermore, clusters C_j for all $j \leq k-2$ have to be present in T' as well as T, because all clusters induced by nodes of rank $t-1$ or lower coincide in these two trees. Cluster C_{k-1}, however, might not be induced by a node in T' if $k-1=t$. Therefore, the first move on FP(T', R) can decrease the rank of the most recent common ancestor of either C_{k-1} or C_k.

We will distinguish two cases depending on whether T and T' are connected by an NNI or a rank move. For each of these we will further distinguish all possible moves between T and T_1. Note that in all figures illustrating possible moves on FP(T, R) and FP(T', R) below, the position of the tree root is irrelevant, so we have positioned roots to simplify our figures.

Case 1. T and T' are connected by an NNI move. So $[(T)_t, (T)_{t+1}]$ is an edge in T—see Fig. 3. Denote the clusters induced by the children of $(T)_t$ by A and B and the cluster induced by the child of $(T)_{t+1}$ that is not $(T)_t$ by C, and assume that the NNI move between T and T' exchanges the subtrees induced by clusters B and C. Additionally, if $(T)_{t+2}$ is the parent of $(T)_{t+1}$ (Cases 1.2 and 1.3), we denote the cluster induced by the child of $(T)_{t+2}$ that is not $(T)_{t+1}$ by D—see Fig. 3.

We now consider all possible moves FindPath can perform to go from T to T_1 that involve a node of rank t or $t+1$, that is, we will consider three intervals in total.

1.1 RNNI move (either type) on interval $[(T)_t, (T)_{t+1}]$. This move has to be the NNI move that is different from the NNI move connecting T and T'. In this case, the cluster $B \cup C$ is built in T_1, as depicted in the bottom of Fig. 3. Hence the first cluster C_k that satisfies the **while** condition of FindPath must contain elements from both B and C but not from A, and the rank of $(C_k)_R$ has to be at most t. But then FindPath applied to T' and R has to decrease the rank of $(C_k)_{T'}$ in its first step implying that $T'_1 = T_1$, so $|\text{FP}(T', R)| = |\text{FP}(T, R)|$. This contradicts our assumption that $|\text{FP}(T', R)| < |\text{FP}(T, R)| - 1$.



(a) Possible initial segments of FP(T, R) **(b)** Possible initial segments of FP(T', R)

Fig. 4 Comparison of paths FP(T, R) and FP(T', R) if T and T' are connected by an NNI move on edge $[(T)_t, T_{t+1}]$ in T. The bottom row displays all possibilities for T_2 and T'_2, depending on the position of cluster C_k that satisfies the **while** condition of FindPath: case C_k intersects B and D is on the left, C_k intersects A and D is in the middle, and C_k intersects C and D is on the right

1.2 NNI move on (edge) interval $[(T)_{t+1}, (T)_{t+2}]$ that swaps the subtrees induced by clusters C and D. This move is shown in Fig. 4a by an arrow from T to the leftmost tree in the middle row. In this case, the first cluster C_k that satisfies the **while** condition of FindPath computing FP(T, R) must intersect D but not C. Additionally, C_k must intersect A, or B, or both of them. Hence, we will consider each of these three cases individually, and demonstrate them in Fig. 4.

1.2.1 C_k intersects A, B, and D but not C. In this case, since we assumed $[(T_1)_t, (T_1)_{t+1}]$ to be an edge in the tree, no move on T_1 can decrease the rank of $(C_k)_{T_1}$. It follows from the proof of Proposition 1 that this can happen only when the subtrees induced by $(C_k)_{T_1}$ and $(C_k)_R$ in the corresponding trees coincide. That is, the **while** condition of FindPath must be false after this first move for all $j \leq k$. This implies that $t = k - 1$ and $C_{k-1} = A \cup B$. But since the rank of $(C_{k-1})_{T'}$ is $t + 1 > k - 1$, C_{k-1} has to be the first cluster for which the **while** condition of FindPath applied to T' and R is met. Hence the first move on FP(T', R) must decrease the rank of $(C_{k-1})_{T'}$ by building the cluster $A \cup B$, in which case $T'_1 = T$. This however contradicts $|\text{FP}(T', R)| < |\text{FP}(T, R)| - 1$.

1.2.2 C_k intersects A and D but not B or C. Starting from T, FindPath exchanges first subtrees induced by clusters C and D and then by B and D. This results in trees T_1 and T_2—see the path leading to the tree in the middle of the bottom row in Fig. 4a. This implies that the rank of $(C_{k-1})_R$ is lower than t,

so the first cluster that satisfies the **while** condition of FindPath applied to T' and R is C_k. Hence, starting from T', FindPath exchanges first subtrees induced by B and D and then by C and D. This results in trees T_1' and T_2'—see the path leading to the tree in the middle of the bottom row in Fig. 4b. It follows that T_2 and T_2' are connected by an RNNI move on the interval $[(T_2)_{t+1}, (T_2)_{t+2}]$ (indicated by dotted edges in the corresponding trees in Fig. 4). This together with the facts that $|\mathrm{FP}(T_2, R)| = |\mathrm{FP}(T, R)| - 2$ and $|\mathrm{FP}(T_2', R)| = |\mathrm{FP}(T', R)| - 2$ contradicts the assumption that $\mathrm{FP}(T, R)$ is of minimal length violating inequality (1).

1.2.3 C_k intersects B and D but not A or C. This case is analogous to the previous one. The two initial segments of $\mathrm{FP}(T, R)$ and $\mathrm{FP}(T', R)$ are the paths leading to the leftmost trees in the bottom row of Fig. 4a and b, respectively. Note that the rank swap leading from T_1' to T_2' is required because the rank of $(C_k)_R$ is at most t as implied by the move leading from T_1 to T_2. The corresponding trees T_2 and T_2' are again RNNI neighbours.

1.3 NNI move on (edge) interval $[(T)_{t+1}, (T)_{t+2}]$ that builds a cluster $C \cup D$ in T_1. This move is shown in Fig. 4a by an arrow from T to the second leftmost tree in the middle row. In this case, C_k intersects C and D but not A or B. And we have the following two possibilities to consider.

1.3.1 The ranks of $(C_k)_{T_1}$ and $(C_k)_R$ coincide. In this case, the previous cluster C_{k-1} of R has to be $A \cup B$. Since $A \cup B$ is not a cluster in T', the first RNNI move on $\mathrm{FP}(T', R)$ builds the cluster $A \cup B$ by swapping subtrees induced by cluster B and C. This move results in $T_1' = T$ contradicting $|\mathrm{FP}(T', R)| < |\mathrm{FP}(T, R)| - 1$.

1.3.2 The rank of $(C_k)_{T_1}$ is strictly higher than that of $(C_k)_R$. In this case, FindPath decreases the rank of $(C_k)_{T_1}$ in the second step. This results in the path from T to the rightmost tree in Fig. 4a. Hence, $\mathrm{FP}(T', R)$ also has to begin with two moves that decrease the rank of $(C_k)_{T'}$ twice, resulting in the rightmost path in Fig. 4b. Similarly to case 1.2.2, we arrive at a contradiction that trees T_2, T_2', and R violate inequality (1) and $|\mathrm{FP}(T_2, R)| < |\mathrm{FP}(T, R)|$.

1.4 Rank move on interval $[(T)_{t+1}, (T)_{t+2}]$. This case is analogous to case 1.3 (see Fig. 5). If the ranks of $(C_k)_{T_1}$ and $(C_k)_R$ coincide then $C_{k-1} = A \cup B$, and applying FindPath to T', R we get $T_1' = T$. If the rank of $(C_k)_{T_1}$ is strictly higher than that of $(C_k)_R$ then FindPath decreases the rank of $(C_k)_{T_1}$ in the second step. Recall that the interval between nodes of rank t and $t+1$ is an edge in both T and T'. Hence, the first two moves on $\mathrm{FP}(T', R)$ decrease the rank of $(C_k)_{T'}$ twice resulting in T_2' which is an RNNI neighbour of T_2 as depicted in Fig. 5. As before, this contradicts our minimality assumption.

1.5 RNNI move (either type) on interval $[(T)_{t-1}, (T)_t]$. In this case $C_k \subseteq A \cup B$ and the rank of $(C_k)_R$ is at most $t-1$. This implies that C_k is the first cluster to satisfy the **while** condition for T' and the first move on $\mathrm{FP}(T', R)$ decreases the rank of $(C_k)_{T'}$ by exchanging the subtrees induced by B and C. This results in $T_1' = T$.

Case 2. T and T' are connected by a rank move. We assume that the rank move is performed on the interval $[(T)_t, (T)_{t+1}]$. Denote the cluster induced by $(T)_t$ by A, the clusters induced by the children of $(T)_t$ by A_1 and A_2, the cluster induced by $(T)_{t+1}$ by B, and the clusters induced by the children of $(T)_{t+1}$ by B_1 and B_2—see Fig. 6.

We again consider all possible moves FindPath can perform to go from T to T_1 that involve a node of rank t or $t+1$.

2.1 Rank move on $[(T)_t, (T)_{t+1}]$. This move results in $T_1 = T'$.

2.2 NNI move on (edge) interval $[(T)_{t+1}, (T)_{t+2}]$. The following two sub-cases are analogous to case 1.3.

2.2.1 $(T)_{t+2}$ is a parent of $(T)_t$. The first move on FP(T, R) builds a cluster $A \cup B_1$ or $A \cup B_2$, and we assume without loss of generality that it is the former, as in Fig. 6. This implies that C_k intersects A and B_1 but not B_2 If the ranks of $(C_k)_{T_1}$ and $(C_k)_R$ coincide then the previous cluster C_{k-1} of R has to be A. Therefore, the first move on FP(T', R) decreases the rank of $(A)_{T'}$, which results in $T_1' = T$. If the rank of $(C_k)_{T_1}$ is strictly higher than that of $(C_k)_R$ then FindPath decreases the rank of $(C_k)_{T_1}$ in the second step. Due to the symmetry we can assume that $C_k \subseteq A_1 \cup B_1$, which implies that the move between T_1 and T_2 exchanges the subtrees induced by A_2 and B_1, as depicted on the left of Fig. 6. $C_k \subseteq A_1 \cup B_1$ implies that the first two moves on FP(T', R) result in a tree T_2' that is an RNNI neighbour of T_2—see Fig. 6. This is a contradiction to the minimality assumption on $|\text{FP}(T, R)|$.

2.2.2 $(T)_{t+2}$ is not a parent of $(T)_t$. In this case, there exists a cluster C induced by the child of $(T)_{t+2}$ which is different from the one that induces B—see Fig. 7. We can assume without loss of generality that $C_k \subseteq C \cup B_1$ and the first move on FP(T, R) builds a new cluster $C \cup B_1$. If the ranks of $(C_k)_{T_1}$ and $(C_k)_R$ coincide then $C_{k-1} = A$, which implies that A is induced by the node of rank t in both T and R. So $T_1' = T$. If the rank of $(C_k)_{T_1}$ is strictly higher than that of $(C_k)_R$ then FindPath decreases the rank of $(C_k)_{T_1}$ in the second step—see Fig. 7. The corresponding first moves on FP(T', R) are shown on the right in Fig. 7, and we again get that T_2 and T_2' are RNNI neighbours.

2.3 Rank move on interval $[(T)_{t+1}, (T)_{t+2}]$. Again, depending on whether or not the ranks of $(C_k)_{T_1}$ and $(C_k)_R$ coincide, we arrive at the conclusion that either $T_1' = T$ or T_2 and T_2' are RNNI neighbours, similarly to case 1.4.

2.4 RNNI move (either type) on interval $[(T)_{t-1}, (T)_t]$. In this case $C_k \subseteq A$ and the first move on FP(T', R) must be a rank swap resulting in $T_1' = T$.

Since all possible cases result in a contradiction, we conclude that inequality (1) is true for all trees, which completes the proof of the theorem. □

We finish this section by showing that no algorithm has strictly lower worst-case time complexity than FindPath. We again assume here that the output of an algorithm for solving RNNI(1)-SP is a list of RNNI moves. Requiring the output to be a list of trees would result in cubic complexity while maintaining the optimality of FindPath.

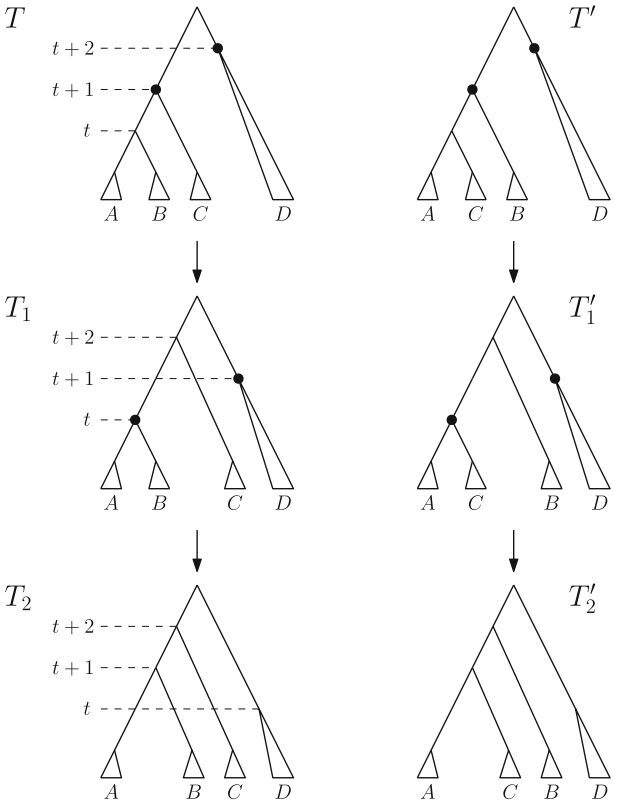


Fig. 5 Comparison of paths FP(T, R) and FP(T', R) if there is an NNI move between T and T' and a rank move on the interval above this edge follows on FP(T, R)

Corollary 1 *The time-complexity of the shortest path problem* RNNI(1)-SP *is* $\Omega(n^2)$.

Proof We prove this by establishing the lower bound on the output size to the problem, that is, the length of a shortest paths.

Consider two "caterpillar" trees $T = [\{a_1, a_2\}, \{a_1, a_2, a_3\}, \ldots, \{a_1, a_2, \ldots, a_n\}]$ and $R = [\{a_1, a_n\}, \{a_1, a_n, a_{n-1}\}, \ldots, \{a_1, a_n, \ldots, a_2\}]$. Applied to these trees FINDPATH executes an NNI move in each of the $n-k-1$ **while** loops (line 3) in every iteration k of the **for** loop (line 2). Hence the length of the output path of FINDPATH is $\sum_{k=1}^{n-2} k = \frac{(n-1)(n-2)}{2}$ and therefore quadratic in n. Theorem 1 then implies that this path is a shortest path. It follows that the worst-case size of the output to RNNI(1)-SP is quadratic. □

4 For what ρ is RNNI(ρ)-SP polynomial?

As we have seen in Sect. 2, the shortest path problem RNNI(1)-SP is solvable in polynomial time. In this section, we will show that a classical result in mathematical

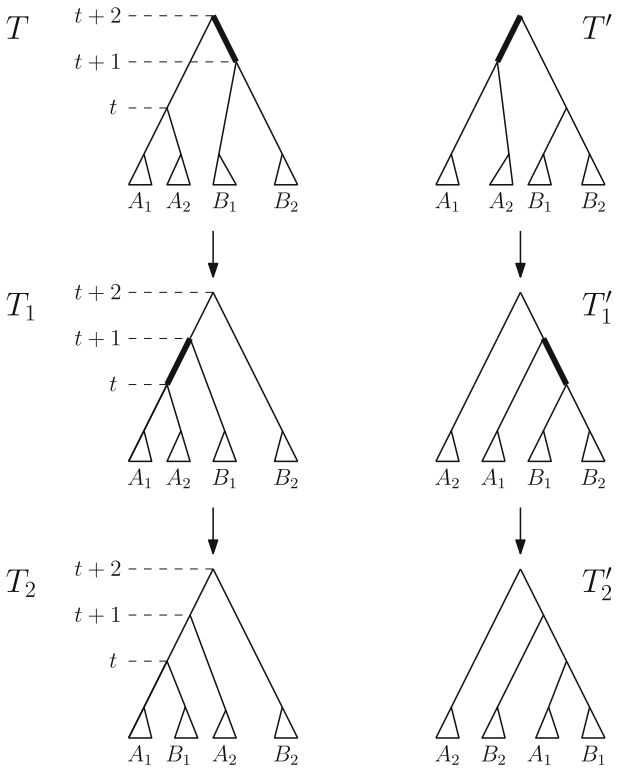


Fig. 6 Rank move between T and T' and possible initial segments of FP(T, R) and FP(T', R) when $[(T)_{t+1}, (T)_{t+2}]$ is an edge. We use notations $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$

phylogenetics implies that RNNI(0)-SP is **NP**-hard. We will also discuss RNNI(ρ)-SP for other values of ρ.

Theorem 2 (DasGupta et al. 2000) RNNI(0)-SP *is* **NP**-*hard.*

Proof Because two trees with the same tree topology but different rankings have distance 0 in RNNI(0), this graph corresponds to a pseudo-metric space. The length of the path required in an instance of RNNI(0)-SP is equal to the minimum number of NNI moves necessary to convert one tree into another tree, as rank moves weigh 0. Therefore, the distance in RNNI(0) equals the NNI distance between trees where the rankings of internal nodes are ignored and NNI moves are allowed on every edge. The corresponding shortest path problem is known to be **NP**-hard (DasGupta et al. 2000). □

In the light of Theorems 1 and 2 the following problem is natural.

Problem 1 Characterise the complexity of RNNI(ρ)-SP in terms of ρ.

This problem is also of applied value. For example, trees might come from an inference method with higher certainty of their branching structure and lower certainty

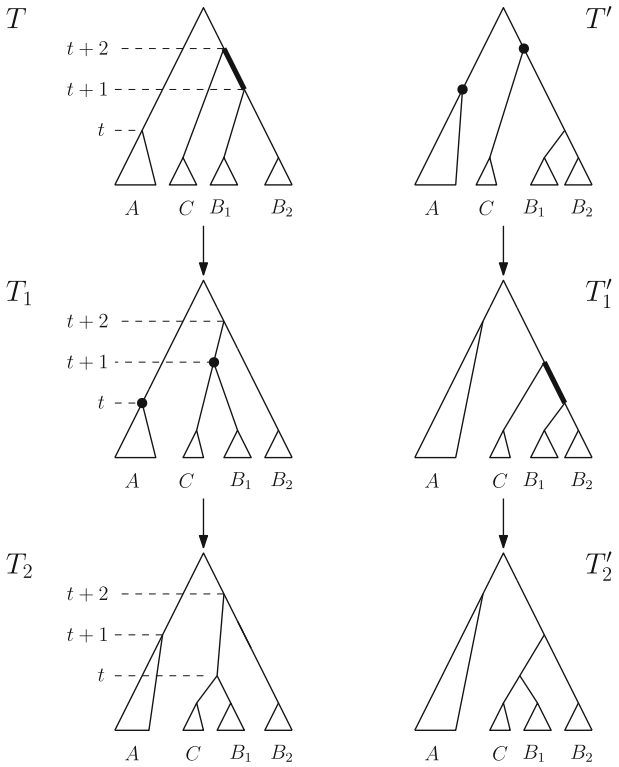


Fig. 7 Comparison of paths FP(T, R) and FP(T', R) if there is a rank move between T and T' and an NNI move on the edge below the corresponding (rank) interval follows on FP(T, R)

of their nodes order. A comparison method for such trees should have higher penalty for NNI changes and lower penalty for rank changes, which in our notations requires $\rho < 1$.

In the rest of this section, we show that the FINDPATH algorithm substantially relies on the fact that the rank move and the NNI move have the same weight in the RNNI graph. This suggests that a non-trivial algorithmic insight is necessary to extend our polynomial complexity result to other values of ρ.

Proposition 2 FINDPATH *does not compute shortest paths in* RNNI(ρ) *for* $\rho \neq 1$.

Proof For $\rho > 1$ a counterexample is given by the following trees (see Fig. 8)

$$T = [\{a_1, a_2\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\}] \text{ and}$$
$$R = [\{a_3, a_4\}, \{a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4\}].$$

Applied to these trees FINDPATH proceeds from T to $[\{a_1, a_2\}, \{a_3, a_4\}, \{a_1, a_2, a_3, a_4\}]$, then to $[\{a_3, a_4\}, \{a_1, a_2\}, \{a_1, a_2, a_3, a_4\}]$, and then to R. This path consists of two NNI moves with one rank move in between them and therefore has weight $2 + \rho$. However, the path from T to $[\{a_2, a_3\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\}]$ to

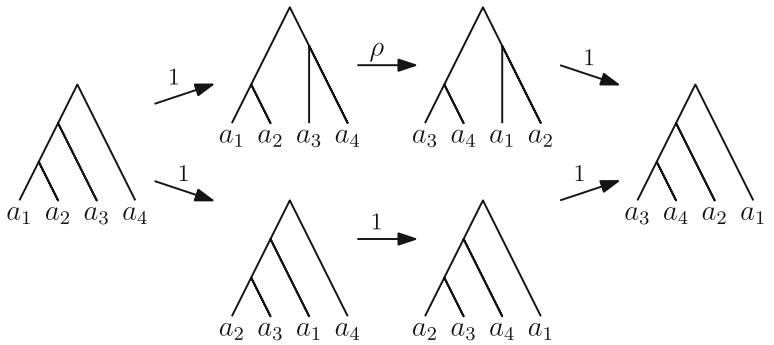


Fig. 8 Path computed by FINDPATH (top) and a shorter path (bottom) for $\rho > 1$

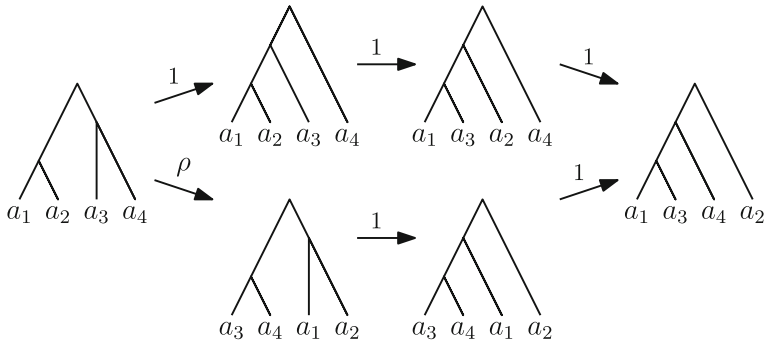


Fig. 9 Path computed by FINDPATH (top) and a shorter path (bottom) for $\rho < 1$

$[\{a_2, a_3\}, \{a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4\}]$ to R consists of three NNI moves and is hence shorter.

For $\rho < 1$ a counterexample is given by the following trees (see Fig. 9)

$$T = [\{a_1, a_2\}, \{a_3, a_4\}, \{a_1, a_2, a_3, a_4\}] \text{ and}$$
$$R = [\{a_1, a_3\}, \{a_1, a_3, a_4\}, \{a_1, a_2, a_3, a_4\}].$$

Applied to these trees FINDPATH proceeds from T to $[\{a_1, a_2\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\}]$, then to $[\{a_1, a_3\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\}]$, and then to R. This path consists of three NNI moves and therefore has weight 3. However, the path from T to $[\{a_3, a_4\}, \{a_1, a_2\}, \{a_1, a_2, a_3, a_4\}]$ to $[\{a_3, a_4\}, \{a_1, a_3, a_4\}, \{a_1, a_2, a_3, a_4\}]$ to R consists of one rank move followed by two NNI moves and is hence shorter. □

5 Additional open problems

The idea utilised by DasGupta et al. (2000) to prove that computing distances in NNI is **NP**-hard stems from a result that shortest paths in NNI do not preserve clusters (Li et al. 1996), that is, sometimes a cluster shared by two trees T and R is shared by

no other tree on any shortest path between T and R. This counter-intuitive property eventually led to the computational hardness result in NNI. Moreover, this property makes little sense biologically as trees clustering the same set of sequences into a subtree should be closer to each other than to a tree that does not have that subtree. Indeed, a shared cluster means that both trees support the hypothesis that this cluster has evolved along a subtree. In light of this biological argument, the **NP**-hardness result can be interpreted as RNNI(ρ)-SP being hard only when the graph RNNI(ρ) is biologically irrelevant. From this paper we know that RNNI(1)-SP can be solved in polynomial time by an algorithm that preserves clusters. This however does not mean that every shortest path in RNNI preserves clusters. The following question is hence natural.

(1) For which values of ρ does RNNI(ρ) have the cluster property? How do those compare to the values of ρ for which RNNI(ρ)-SP is efficient?

Other natural questions that arise in the context of our results are the following.

(2) The questions we have considered for ranked NNI can be studied in other rearrangement-based graphs on leaf-labelled trees, such as the ranked SPR graph and the ranked TBR graph (Semple and Steel 2003). What is the complexity of the shortest path problem there?

(3) Can our results be used to establish whether the problem of computing geodesics between trees with real-valued node heights is polynomial-time solvable? This geodesic metric space is called t-space and an efficient algorithm for computing geodesics in t-space would be of importance for applications (Gavryushkin and Drummond 2016).

References

Allen BL, Steel M (2001) Subtree transfer operations and their induced metrics on evolutionary trees. Ann Comb 5(1):1–15

Alves JM, Sonia P-L, Manuel C-TJ, David P (2019) Rapid evolution and biogeographic spread in a colorectal cancer. Nat Commun 10(1):1–7

Bansal MS, Gordon BJ, Oliver E, David F-B (2010) Robinson–Foulds supertrees. Algorithms Mol Biol 5(February):18

Bordewich M, Semple C (2005) On the computational complexity of the rooted subtree prune and regraft distance. Ann Comb 8(4):409–423

Bouckaert RR, Bowern C, Atkinson QD (2018) The origin and expansion of Pama–Nyungan languages across Australia. Nat Ecol Evol 2(4):741–749

Bouckaert R, Vaughan TG, Barido-Sottani J, Duchene S, Fourment M, Gavryushkina A, Heled J, Jones G, Kuhnert D et al (2019) BEAST 2.5: an advanced software platform for Bayesian evolutionary analysis. PLoS Comput Biol 15(4):e1006650

Collienne L, Elmes K, Berling L, Gavryushkin A (2019) RNNI code. https://github.com/bioDS/treeOclock
DasGupta B, He X, Jiang T, Li M, Tromp J (1999) On the linear-cost subtree-transfer distance between phylogenetic trees. Algorithmica 25(2):176–195
DasGupta B, He X, Jiang T , Li M, Tromp J, Zhang L (2000) On computing the nearest neighbor interchange distance. In: Discrete mathematical problems with medical applications: DIMACS workshop discrete mathematical problems with medical applications, December 8–10, 1999, vol 55. DIMACS Center, p 19
Downey RG, Fellows MR (2013) Fundamentals of parameterized complexity. Springer, London
Gavryushkin A, Drummond AJ (2016) The space of ultrametric phylogenetic trees. J Theor Biol 403(August):197–208
Gavryushkina A, Welch D, Stadler T, Drummond AJ (2014) Bayesian inference of sampled ancestor trees for epidemiology and fossil calibration. PLoS Comput Biol 10(12):e1003919
Gavryushkin A, Whidden C, Matsen FA (2018) The combinatorics of discrete time-trees: theory and open problems. J Math Biol 76(5):1101–1121
Gray RD, Drummond AJ, Greenhill SJ (2009) Language phylogenies reveal expansion pulses and pauses in Pacific settlement. Science 323(5913):479–483
Guindon S, Dufayard J-F, Lefort V, Anisimova M, Hordijk W, Gascuel O (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. Syst Biol 59(3):307–321
Hein J, Jiang T, Wang L, Zhang K (1996) On the complexity of comparing evolutionary trees. Discrete Appl Math 71(1):153–169
Hickey G, Dehne F, Rau-Chaplin A, Blouin C (2008) SPR distance computation for unrooted trees. Evol Bioinform Online 4:17–27
Li M, Tromp J, Zhang L (1996) Some notes on the nearest neighbour interchange distance. In: Cai JY, Wong CK (eds) Computing and combinatorics. COCOON 1996. Lecture Notes in Computer Science, vol 1090. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61332-3_168
Lote H, Spiteri I, Ermini L, Vatsiou A, Roy A, McDonald A, Maka N, Balsitis M, Bose N et al (2017) Carbon dating cancer: defining the chronology of metastatic progression in colorectal cancer. Ann Oncol 28(6):1243–1249
McMorris FR, Steel MA (1994) The complexity of the median procedure for binary trees. In: Diday E, Lechevallier Y, Schader M, Bertrand P, Burtschy B (eds) New approaches in classification and data analysis. Studies in classification, data analysis, and knowledge organization. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-51175-2_14
Nguyen L-T, Schmidt HA, von Haeseler A, Quang MB (2015) IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol Biol Evol 32(1):268–274
Robinson DF, Foulds LR (1981) Comparison of phylogenetic trees. Math Biosci 53(1):131–147
Ronquist F, Huelsenbeck JP (2003) MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19(12):1572–1574
Semple C, Steel M (2003) Phylogenetics. Oxford University Press, Oxford
Singer J, Kuipers J, Jahn K, Beerenwinkel N (2018) Single-cell mutation identification via phylogenetic inference. Nat Commun 9(1):5144
Stamatakis A (2006) RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics 22(21):2688–2690
Suchard MA, Lemey P, Baele G, Ayres DL, Drummond AJ, Rambaut A (2018) Bayesian phylogenetic and phylodynamic data integration using BEAST 1.10. Virus Evol 4(1):vey016
Tamura K, Peterson D, Peterson N, Stecher G, Nei M, Kumar S (2011) MEGA5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. Mol Biol Evol 28(10):2731–2739
Whidden C, Matsen FA (2017) Ricci–Ollivier curvature of the rooted phylogenetic subtree–prune–regraft graph. Theoret Comput Sci 699:1–20. https://doi.org/10.1016/j.tcs.2017.02.006
Whidden C, Matsen FA (2018) Calculating the unrooted subtree prune-and-regraft distance. IEEE ACM Trans Comput Biol Bioinform 16:898–911
Whidden C, Beiko RG , Zeh N (2010) Fast FPT algorithms for computing rooted agreement forests: theory and experiments. In: Experimental algorithms. Lecture notes in computer science , pp 141–153
Whidden C, Zeh N, Beiko RG (2014) Supertrees based on the subtree prune-and-regraft distance. Syst Biol 63(4):566–581

Ypma RJF, van Ballegooijen WM, Wallinga J (2013) Relating phylogenetic trees to transmission trees of infectious disease outbreaks. Genetics 195(3):1055–1062