ISSN 0032-9460, Problems of Information Transmission, 2026, Vol. 62, No. 2, pp. 185–201. © Pleiades Publishing, Inc., 2026.

LARGE SYSTEMS

Discrete Evolution as a Method for Computing the MAF Distance between Trees

K. Yu. Gorbunov[a,*] **and V. A. Lyubetsky**[a,**]

[a] *Kharkevich Institute for Information Transmission Problems of the Russian Academy of Sciences, Moscow, Russia*
e-mail: *gorbunov@iitp.ru, **lyubetsk@iitp.ru

Received March 27, 2026; revised September 14, 2026; accepted September 22, 2026

Abstract—We present a new quadratic-time algorithm that computes an embedding of one rooted binary tree into another. Using this algorithm, we approximately compute the well-known MAF distance between two trees. We provide a proof sketch demonstrating that the distance thus computed exceeds the exact MAF distance by a factor of at most two.

Key words: discrete optimization algorithm, MAF distance, quadratic-complexity algorithm, algorithm accuracy, tree embedding.

DOI: 10.1134/S0032946026020043

1. INTRODUCTION AND PROBLEM STATEMENT

The concept of distance between mathematical objects and methods for its efficient computation remain crucial and long-studied topics. These concepts are highly relevant to both fundamental and applied research. Such applications include defining and computing the distance (similarity, or commonality) between biological species, cells, genes, and genomes. These problems are closely related to computing the distance between trees investigated in this work, which is essentially equivalent to the well-known MAF distance.

The distance between two character sequences (referred to as *words*) over a given alphabet is often computed using word alignment or, equivalently, via the Levenshtein edit distance. Trees, acyclic graphs, and other graph classes provide a natural and widely used generalization of words; for example, graphs whose vertices have degrees not exceeding two (i.e., sets of chains and cycles) are highly important in many respects, see, e.g., [1]. Hereafter, we consider trees where an incoming edge enters the root; such a tree is called rooted. References [2, 3] proposed an exact quadratic algorithm for the alignment of rooted binary trees whose vertices (or leaves only) are labeled. They also introduced a general model for the evolution of one tree relative to another. Notably, that algorithm computes a distance metric that differs from the MAF distance.

In practical applications, various methods are used to define the distance between two given trees T_1 and T_2, including metrics that evaluate the similarity of large isomorphic parts within T_1 and T_2. Such a distance can be computed using two forests that maximize the average tree size while minimizing the total number of trees. Specifically, these forests are families of subtrees generated from T_1 and T_2, respectively, that are pairwise (i.e., bijectively) isomorphic. This metric is called the MAF distance and is considered one of the most biologically meaningful and relevant measures, as shown, in particular, in [4–6]. Defined via a forest of induced subtrees, the MAF distance is introduced, for example, in [4], which also outlines its historical background and provides further references. In this paper, we consider the *MAF distance from the perspective of evolution* and *without any restrictions on T_1 and T_2*.

Computing the MAF distance is an NP-hard problem [7, 8]; therefore, developing an algorithm for its computation requires an approximation approach. Note that FPT algorithms [9] can solve this problem exactly for small distances. A primary reason for the significant interest in this problem lies in the connection between the MAF distance and the well-studied rSPR distance [8]. References [4, 10] described a quadratic-time algorithm (proven in [4], Appendix A) for computing the MAF distance with a multiplicative approximation factor of 2. The same result was obtained by the authors in the conference version [10] of their paper [4], and independently with cubic time complexity in [11]. However, the description of the algorithm in [4] is highly complex, and its correctness proof, which employs linear programming duality, is even more intricate. The authors of [4] themselves formulate both simplifying their algorithm and streamlining the error bound proof as open problems. In the present work, we provide a relatively simple and entirely different quadratic algorithm along with a proof of its correctness achieving the same error bound. Instead of using agreement forests, duality, or linear programming, we employ dynamic programming and model the evolution of one tree relative to another as discrete time. All details of the proof are provided in [12].

Thus, we consider *binary* trees whose leaves are uniquely (i.e., bijectively) labeled with the same set of symbols across all trees. The roots of all trees are positioned at the top, and the leaves are at the bottom. A partial order on the vertices and edges of a tree is defined by the descendant-ancestor relation and is denoted by "$<$" (where a descendant lies "below" an ancestor). For an edge e of a tree, let e_+ denote its *tail* (closer to the root) and e_- denote its *head* (further from the root). A *clade* K_v (respectively, K_e) of a tree is the set of all leaves located below a given vertex v (respectively, below an edge e). The subtree rooted at v with this leaf set is denoted by T_v (respectively, T_e). In the original tree T, the edge entering a vertex $v \in T$ from above is called the *root edge* of the subtree T_v (and of the clade K_v), assuming that such an edge also exists at the root of the original tree, with its tail called the *superroot*.

For an arbitrary leaf set $L \subseteq T$, the *induced* $L-$*tree* T' (which is not necessarily a subtree of T) is the restriction of T to the leaf set L, rooted at the lowest common ancestor of L, with the following modification: all vertices of degree 2 are suppressed, and their incoming and outgoing edges are merged. We treat this restriction T' as an independent tree, which may not constitute a subtree of T. Let S denote a collection of clades in the tree T, referred to as S-*clades*. A *strict leaf* $a \in K \in S \in T$ is defined as a leaf that does not belong to any clade $K' \in S$ strictly contained in K (i.e., $K' \subsetneqq K$). Let K^S denote the set of strict leaves in the clade $K \in S \in T$. Here and below, the notation $S \in T$ means that S is a set of clades in T. The concept of a strict leaf depends on both the clade K and the collection S of clades in the same tree T.

Let T_1 and T_2 be two binary trees whose leaves share a common, unique label set. Consider a *partial bijection* S' between the collections of all clades $\mathcal{T}_1$ in T_1 and $\mathcal{T}_2$ in T_2, denoted by $S' \colon \mathcal{T}_1 \to \mathcal{T}_2$. Let $K_1^{S'1}$ be the set of strict leaves for a clade $K_1 \in \mathcal{T}_1$ that appears as a left element (domain) in S'. The set $K_2^{S'2}$ is defined analogously for a clade $K_2 \in \mathcal{T}_2$ that appears as a right element (codomain) in S'. The partial bijection S' is called *consistent* if the following conditions hold:

S-1. If $(K_1, K_2) \in S'$, then $K_1^{S'1} =' K_2^{S'2} \neq \varnothing$, where $='$ denotes the equality of leaf label sets.

S-2. Every leaf in T_1 is strict in some clade K_1 that forms the left element of a pair $(K_1, K_2) \in S'$ (by property S-1, the symmetric statement holds for leaves in T_2). Hereafter, leaves and their labels are identified.

S-3. If $(K_1, K_2) \in S'$, then the induced $K_1^{S'1}$- and $K_2^{S'2}$-trees (of T_1 and T_2, respectively) are isomorphic preserving leaf labels.

Figure 1 illustrates an example of a consistent partial bijection $S'(f)$ between the clades of trees T_1 and T_2, induced by a given embedding $f \colon T_1 \to T_2$ (see Section 2 and Lemma 1 in Section 3).

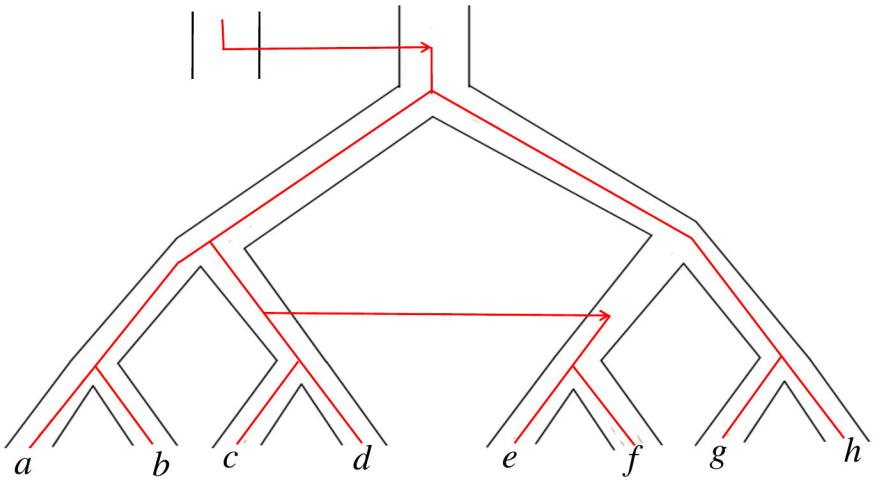


Fig. 1. The embedding $f\colon T_1 \to T_2$ for the trees $T_1 = (((a,b),((c,d),(e,f))),(g,h))$ and $T_2 = (((a,b),(c,d)),((e,f),(g,h)))$. The tree T_2 is represented by tubes with black boundaries, while the tree T_1 is depicted by edges drawn inside T_2. The consistent partial bijection $S'(f)$ corresponding to this embedding f, given by $(\{a,b,c,d,e,f,g,h\},\{a,b,c,d,e,f,g,h\})$ and $(\{e,f\},\{e,f\})$, is constructed according to Lemma 1. For the clades in the domain (left side) of the bijection, the strict leaves in T_1 are $\{a,b,c,d,g,h\}$ for the clade $\{a,b,c,d,e,f,g,h\}$ and $\{e,f\}$ for the clade $\{e,f\}$; symmetrically, the same leaf sets are strict for the corresponding clades in T_2. Two transfers under f are indicated by arrows; the embedding f itself is conflict-free (see the definition below).

The figure also displays the sets of strict leaves for two left clades in T_1 and their two corresponding right clades in T_2 under $S'(f)$. Here, the collection S comprises either all clades, all left clades, or all right clades from $S'(f)$.

The *distance* $R(T_1,T_2)$ between trees T_1 and T_2 is defined as the *minimum cardinality* of a consistent partial bijection $S^\star\colon \mathcal{T}_1 \to \mathcal{T}_2$ decreased by one. Everywhere below, this optimal bijection is denoted by $S^\star$. As usual, the cardinality of a set is denoted by vertical bars, e.g., $|S^\star|$. The cardinality $|S'|$ of any consistent partial bijection S' is inversely proportional to the average size of the corresponding induced trees; hence, minimizing the cardinality corresponds to maximizing the average size, and vice versa.

The equivalence between our definition of a consistent bijection and the standard definition of an agreement forest [4] can be easily verified and is omitted here.

Thus, the objective is to approximately compute the distance $R(T_1,T_2)$ for the given trees T_1 and T_2, and to construct a corresponding consistent partial bijection, which may differ from $S^\star$. This bijection identifies isomorphic induced trees for T_1 and T_2 by property S-3. We note that existing methods for computing the MAF distance typically rely on the isomorphism between the left and right forests obtained from the original trees T_1 and T_2, respectively (see [4–6]); in contrast, our approach completely obviates the concept of a forest.

2. REDUCTION OF THE MAF DISTANCE PROBLEM TO THE EMBEDDING PROBLEM

As in [3], it is convenient to refer to the edges in T_2 as *tubes*, conceptualizing the evolution of the graph T_1 as occurring inside them (Figs. 1 and 2). We append a *root edge* to T_1 and a *root tube* to T_2, both entering their respective roots from above. The upper endpoint of the root edge in T_1 is called the *superroot*. In T_2, we also introduce an *external tube* that has no child tubes and is assumed to be positioned above all other tubes in T_2.

We consider consistent partial bijections $S'\colon \mathcal{T}_1 \to \mathcal{T}_2$ induced by *embeddings* $f\colon T_1 \to T_2$. These embeddings are also of independent interest, as they model the evolution of T_1 relative to T_2.

Let us define an embedding $f\colon T_1 \to T_2$ via the concept of an *event*. Under f, the superroot in T_1 is located in the external tube where the root edge originates (Fig. 1). Proceeding top-down, each edge $e \in T_1$ originates or continues within a tube $t \in T_2$ (where t is neither the external tube nor a leaf tube in items 1 and 2 below). Under f, the edge e within the tube t undergoes one of the following events:

1. Reach the nearest bifurcation in T_2 and split into distinct child tubes (*speciation*), Fig. 2(1);
2. Pass the bifurcation, continuing into only one of the two child tubes (*loss in* t_-), Fig. 2(2);
3. Transfer from the tube t to any non-external tube t' (*transfer*), followed by a subsequent event, such as a speciation or a transition to a leaf, Figs. 2(3), 2(4);
4. Split within the tube t multiple times (*duplication*), after which all resulting edges are transferred, except possibly for one that reaches the nearest bifurcation (or leaf) in T_2 to undergo a speciation or a loss, Fig. 2(5);
5. If e_- is a leaf, reach a leaf t_- sharing the same label as e_- (*basis pair*). In this case, e and t themselves are called *leaf elements*, Fig. 2(6).

Below, we further refine the categorization of these events by distinguishing between specific types.

Thus, each *event* is associated with a pair (e, t) (Fig. 2).

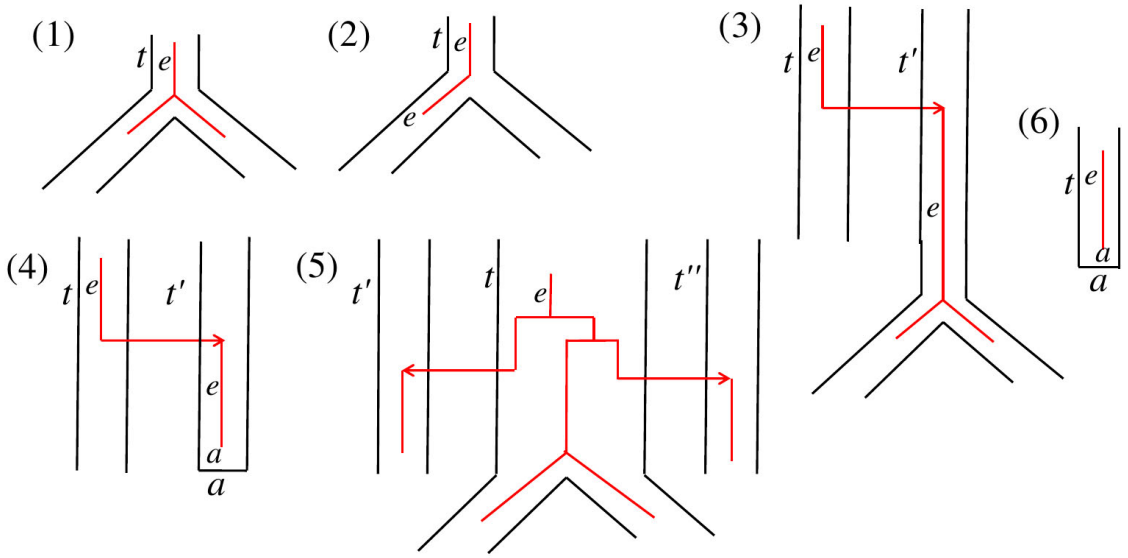


Fig. 2. Six primary events in an embedding $f\colon T_1 \to T_2$ of a tree T_1 into T_2. (1) Speciation of an edge e at t_-. (2) Loss of an edge e at t_-. In this and all subsequent figures, a bend in an edge $e \in T_1$ without a bifurcation denotes the continuation of the same edge e. (3) Transfer of an edge e from a tube t to a tube t' followed by speciation. (4) Transfer of an edge e from a tube t to a tube t' followed by arrival at a leaf. (5) Duplication of an edge e and its descendant edges within a tube t (here, two duplications), where at most one descendant of e exits t through t_-, while all other descendants (if any) are transferred. The figure illustrates two resulting descendants being transferred, and one exiting t via speciation. (6) A leaf edge e arriving at a leaf $e_- \in T_1$ that shares the same label as the leaf $t_- \in T_2$; the label itself is denoted by a.

Figure 1 illustrates an example of an embedding with two transfers, the upper of which originates in the external tube. The formal definition of an embedding f as a mapping from the vertices of T_1 to the vertices and tubes of T_2, along with the formalization of the aforementioned events, straightforwardly follows from this description and is omitted here. Hereafter, the location of a vertex $v \in T_1$ within T_2 is denoted by $f(v)$ and is called the *image* of the vertex v.

We say that two edges e_1 and e_2 of T_1 *conflict* at a vertex t_- if they both exit t, terminating at t_- or below. An embedding without any conflicts is called *conflict-free*. For a given embedding $f\colon T_1 \to T_2$, an edge e within a tube t is called a *pre-transfer edge* if e is not transferred, $f(e_-) = t$, but no duplicated descendant of e within t exits t (though duplications themselves may occur), Fig. 3(1). A speciation of an edge e at t_- is called *transfer-associated* if at least one of its two child edges is transferred from a child tube of t or acts as a pre-transfer edge within it.

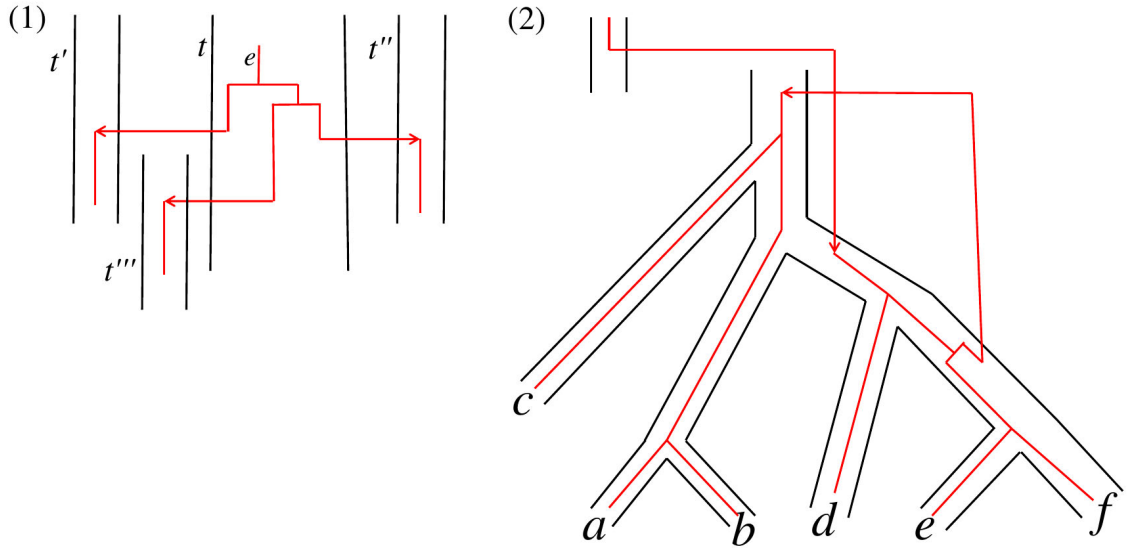


Fig. 3. (1) All duplicated descendants of the edge e are transferred; the edge e acts as a pre-transfer edge. (2) An embedding of the tree $T_1 = (d, ((e, f), (c, (a, b))))$ into $T_2 = (c, ((a, b), (d, (e, f))))$, corresponding to the bijection $S'(f)$ mapping $\{a, b, c, d, e, f\}$ to $\{d, e, f\}$, and $\{a, b, c\}$ to $\{a, b, c, d, e, f\}$.

An edge $e \in T_1$ is called a *dead end* if every path from e_- to a leaf contains a transfer, unless e is a pre-transfer edge and has no losses. An embedding that contains no dead-end edges is called *dead-end-free*.

With each dead-end edge e, we associate a collection D_e of pairwise incomparable non-dead-end edges that undergo a transfer, satisfying the following property: every path from e_- to any leaf contains an edge from D_e. This collection D_e is called *separating* for e (as it separates e_- from the leaves); the subsequent constructions are independent of the choice of D_e.

For any embedding $f\colon T_1 \to T_2$, its *cost* $c(f)$ is defined as

$$c(f) = A(f) + B(f) + C(f),$$

where $A(f)$ is the number of transfers, $B(f)$ is the number of losses (specifically, the number of pairs consisting of an edge and a vertex at which a loss occurs), and $C(f)$ is the number of transfer-associated speciations under f.

Outline of the algorithm:

(1) We construct an embedding $F\colon T_1 \to T_2$ (which is subsequently shown to minimize the cost $c(F)$).

(2) We establish that the constructed embedding F is dead-end-free (Lemma 2).

(3) Beginning with F, we iteratively eliminate conflicts without introducing new ones, which sequentially increases the number of transfers. This yields a sequence

$$F \le F_1 \le \ldots \le F^{\#},$$

where $\le$ denotes the resolution of a single conflict via the addition of one transfer. The terminal embedding $F^{\#}\colon T_1 \to T_2$ is conflict-free and remains dead-end-free, though its cost $c(F^{\#})$ is not necessarily minimal. Nevertheless, the number of transfers $A(F^{\#})$ remains bounded by $A(F)+B(F)+C(F)$ of the initial embedding F. The algorithm outputs the bijection $S'(F^{\#})$ constructed according to Lemma 1, along with its cardinality.

(4) In Theorem 1, leveraging Lemmas 3–9, we prove the correctness of the algorithm according to the following proof outline:

(4.1) Lemmas 3–5 imply that the embedding $F^{\#}$ is both conflict-free and dead-end-free.

(4.2) Following Lemma 5, we demonstrate that the events omitted during the construction of F (specifically, events 7–12 in the list provided below) can be replaced by events 1–6 without increasing the embedding cost or the number of transfers.

(4.3) Lemmas 6 and 7 guarantee the correctness of the first stage of the algorithm, which constructs the minimum-cost embedding $F\colon T_1 \to T_2$. By virtue of step **(4.2)**, this cost is minimal among all valid embeddings, including those featuring events 7–12.

(4.4) Lemma 8 establishes the upper bound

$$A(F^{\#}) \le A(F) + B(F) + C(F).$$

(4.5) Let $F^{\star}$ denote a conflict-free embedding that minimizes the number of transfers. Lemma 9 proves that

$$B(F^{\star}) + C(F^{\star}) < A(F^{\star}).$$

(4.6) From Lemmas 8 and 9 and the minimality of $c(F)$, we obtain

$$A(F^{\#}) \le 2A(F^{\star}) - 1;$$

hence,

$$|S'(F^{\#})| - 1 \le 2(|S^{\star}| - 1).$$

Thus, $S'(F^{\#})$ yields the desired approximation to the optimal bijection $S^{\star}$ within the guaranteed multiplicative error bound.

3. CONNECTION BETWEEN A CONSISTENT CLADE BIJECTION AND TREE EMBEDDING

Lemma 1 implies that, up to a linear-time transformation, the original problem of computing the MAF distance $R(T_1, T_2)$ is equivalent to constructing a conflict-free and dead-end-free embedding $f\colon T_1 \to T_2$ that minimizes the number of transfers among all conflict-free embeddings, yielding $|S'(f)| = |S^{\star}|$.

Lemma 1. *Given a conflict-free and dead-end-free embedding $f\colon T_1 \to T_2$ with A transfers, a consistent partial bijection $S'(f)\colon \mathcal{T}_1 \to \mathcal{T}_2$ between the clades of T_1 and T_2 can be constructed in linear time, such that $|S'(f)| = A$. Conversely, given any consistent partial bijection S', a conflict-free and dead-end-free embedding $f\colon T_1 \to T_2$ can be constructed with the same time complexity, featuring exactly $|S'(f)| = |S'|$ transfers.*

Proof sketch. Given an embedding f, for each transfer of an edge e into a tube t, we include the pair (K_e, K_t) in S' (see Figs. 1 and 3(2)). This collection S' forms a bijection because any edge $e \in T_1$ can be transferred into at most one tube, and in the absence of dead-end edges, two transfers into the same tube t would create a conflict at t_-. Conditions S-1, S-2, and S-3 straightforwardly follow from this construction, and the details are omitted here.

We now address the second statement of the lemma. Given a consistent partial bijection S', we construct the embedding f from it. As an illustration, Fig. 3(2) shows the embedding of the tree

$$T_1 = (d, ((e, f), (c, (a, b))))$$

into

$$T_2 = (c, ((a, b), (d, (e, f)))),$$

corresponding to the bijection $S'(f)$ that mapping $\{a, b, c, d, e, f\}$ to $\{d, e, f\}$ and $\{a, b, c\}$ to $\{a, b, c, d, e, f\}$. After duplicating the root edge of T_1 within the external tube multiple times, we transfer the resulting edges e into tubes t, where $(K_e, K_t) \in S'$ and K_e is maximal in S'_1 (i.e., not strictly contained in any other clade of S'_1). Next, for each resulting pair (e, t), we independently construct an embedding $T_e \to T_2$ such that the superroot of T_e is mapped to t. At each step, we apply one of the events to the edge e, replacing the pair (e, t) with one or more new pairs, which are subsequently processed independently. The details of this construction are omitted here.

If f contains dead-end edges, we eliminate them using the procedure described at the end of Section 4. This process preserves the conflict-free property without altering the number of transfers. $\triangle$

4. DESCRIPTION OF THE ALGORITHM

First stage of the algorithm. We assign a unit cost to each transfer, loss, and transfer-associated speciation event in any embedding f, while the costs of all other events are set to zero. Thus, the *embedding cost* $c(f)$ equals the total number of its transfers $A(f)$, losses $B(f)$, and transfer-associated speciations $C(f)$, i.e.,

$$c(f) = A(f) + B(f) + C(f).$$

In principle, alternative weights for transfers, losses, and transfer-associated speciations could be considered.

1. We construct an *embedding* $F\colon T_1 \to T_2$ (in Section 5, we show that F minimizes the cost $c(f)$, though it is not necessarily conflict-free). For a fixed edge $e \in T_1$ (up to the root edge e_0), we iterate over all tubes $t \in T_2$ (up to the external tube t_0), assigning to each pair (e, t) one of the directional events d illustrated in Figs. 4(1)–4(7). A pair (e, t) associated with such an event d and a direction arrow takes the form $d(e \downarrow t)$ or $d(e \uparrow t)$, which we refer to as a *formula*. We compute the cost $c(\cdot)$ of the formula and link it to the corresponding *expression* (the sum of formulas) from which this cost is derived. This expression contains only lower-order formulas, implementing the forward pass of dynamic programming. Subsequently, the backward pass of dynamic programming yields the *minimal arrangement* of formulas.

This formulation implies that the pair (e, t) occupies one of the states defined by the respective formula.

We compute the costs $c(\cdot)$ as follows. For a basis pair (e, t), we set

$$c(d(e \downarrow t)) = 0,$$

where d denotes a label match. If (e,t) is a leaf pair but not a basis pair, we set

$$c(d(e\downarrow t))=\infty.$$

Otherwise, for $d(e\downarrow t)$ and $d(e\uparrow t)$, the costs are computed recursively. For $d_{\min}(e\downarrow t)$, we have (Figs. 4(4)–4(7)):

$$\begin{aligned}c(d_{\min}(e\downarrow t))=\min\Big\{&c(d_{\min}(e_1\downarrow t_1))+c(d_{\min}(e_2\downarrow t_2)),\\&c(d_{\min}(e_1\downarrow t_2))+c(d_{\min}(e_2\downarrow t_1)),\\&c(d_{\min}(e_1\downarrow t_1))+c(d_{\min}(e_2\uparrow t_2))+1,\\&c(d_{\min}(e_1\uparrow t_1))+c(d_{\min}(e_2\downarrow t_2))+1,\\&c(d_{\min}(e_1\downarrow t_2))+c(d_{\min}(e_2\uparrow t_1))+1,\\&c(d_{\min}(e_1\uparrow t_2))+c(d_{\min}(e_2\downarrow t_1))+1,\\&c(d_{\min}(e\downarrow t_1))+1,\quad c(d_{\min}(e\downarrow t_2))+1,\\&c(d_{\min}(e_1\downarrow t))+c(d_{\min}(e_2\uparrow t)),\\&c(d_{\min}(e_1\uparrow t))+c(d_{\min}(e_2\downarrow t))\Big\}.\end{aligned}$$

For a non-leaf edge e, we compute (Figs. 4(2)–4(3)):

$$\begin{aligned}c(d_{\min}(e\uparrow t))=\min\Big\{&c(d_{\min}(e_1\uparrow t))+c(d_{\min}(e_2\uparrow t)),\\&\min_{t'}\{c(d_{\min}(e_1\downarrow t'_1))+c(d_{\min}(e_2\downarrow t'_2)),\\&\qquad c(d_{\min}(e_1\downarrow t'_2))+c(d_{\min}(e_2\downarrow t'_1))\}+1\Big\}.\end{aligned}$$

If either e or t lacks the child edges indicated in a given sum, that sum is assumed to be ∞. For a leaf edge e, we set

$$c(d_{\min}(e\uparrow t))=1.$$

As is standard in dynamic programming, computing $c(\cdot)$ establishes a *pointer* from a formula to the specific expression (the minimizing sum of formulas) from which it is derived. Here, $d_{\min}$ denotes the specific event identified by this pointer, ensuring that $d_{\min}$ precisely corresponds to one of the minimizing sums.

The internal minimum over t' required to compute $c(d_{\min}(e\uparrow t))$ is determined prior to evaluating the costs $c(d_{\min}(e\downarrow t))$ and $c(d_{\min}(e\uparrow t))$. Specifically, for a non-leaf edge e, we define:

$$t_e=\arg\min\Big\{\min_{t'}\{c(d_{\min}(e_1\downarrow t'_1))+c(d_{\min}(e_2\downarrow t'_2)),\,c(d_{\min}(e_1\downarrow t'_2))+c(d_{\min}(e_2\downarrow t'_1))\}\Big\},$$

and if e is a leaf edge, we define t_e as the leaf tube t sharing the same label as e. Conceptually, t_e represents the optimal tube for the transfer of e.

Upon reaching the pair (e_0,t_0) consisting of the root edge and the external tube during the forward pass, we construct the desired embedding F during the backward pass. This process starts from $d_{\min}(e_0\uparrow t_0)$ evaluated using the same equations, and proceeds by following the pointers. For the current formula $d_{\min}(e\uparrow t)$ or $d_{\min}(e\downarrow t)$, we follow the associated pointer. If the pointer targets the scenario illustrated in Fig. 4(4) or 4(5), then $F(e_-)=t_-$ and we move to the corresponding formulas $d_{\min}(e_1\downarrow t_1)$ and $d_{\min}(e_2\downarrow t_2)$, or $d_{\min}(e_1\downarrow t_1)$ and $d_{\min}(e_2\uparrow t_2)$, respectively. If the pointer targets Fig. 4(2) or 4(7), then $F(e_-)=t$ and we proceed to the formulas $d_{\min}(e_1\uparrow t)$ and $d_{\min}(e_2\uparrow t)$, or $d_{\min}(e_1\downarrow t)$ and $d_{\min}(e_2\uparrow t)$. If the pointer targets Fig. 4(6), we proceed to the formula $d_{\min}(e\downarrow t_1)$, leaving $F(e_-)$ temporarily undefined. If the pointer targets Fig. 4(3),

then $F(e_-) = (t_e)_-$; if e is not a leaf edge, we proceed according to Fig. 4(3) to the formulas $d_{\min}(e_1 \downarrow (t_e)_1)$ and $d_{\min}(e_2 \downarrow (t_e)_2)$, or to $d_{\min}(e_1 \downarrow (t_e)_2)$ and $d_{\min}(e_2 \downarrow (t_e)_1)$ (where $(t_e)_1$ and $(t_e)_2$ are the child tubes of t_e), otherwise we terminate at the basis pair. Finally, if the pointer targets Fig. 4(1), then $F(e_-) = t_-$.

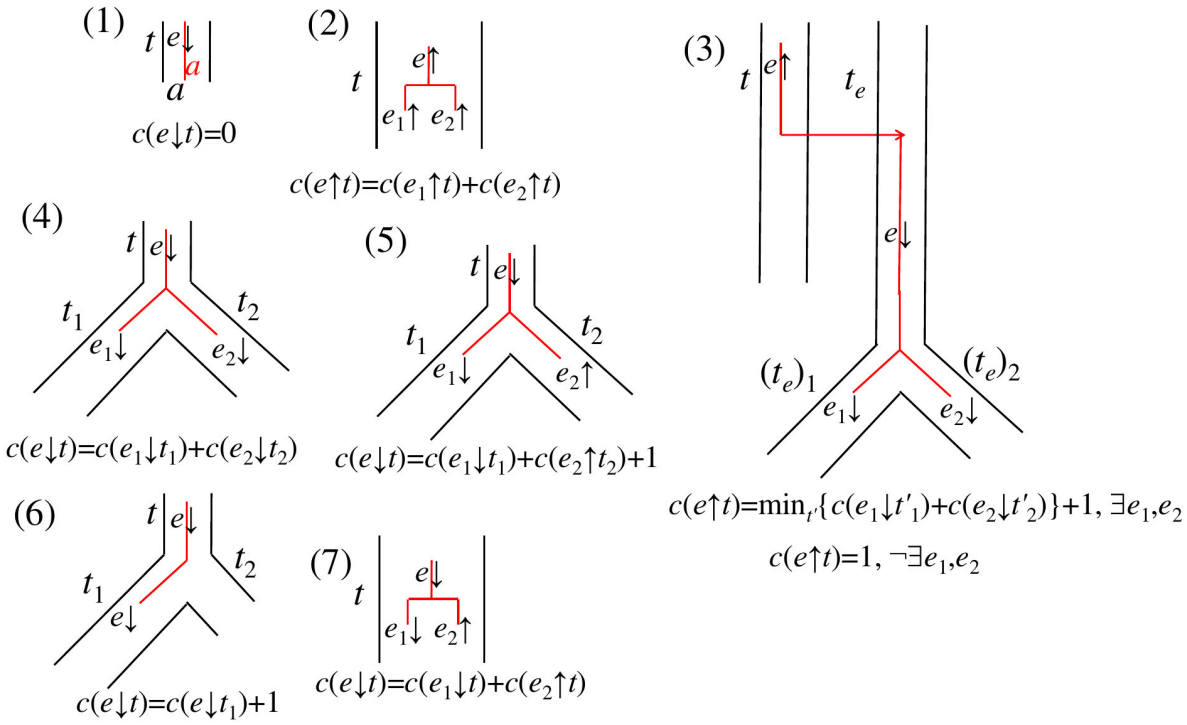


Fig. 4. Variants in the construction of the embedding F (here, the event d is omitted in the formulas): (1) label match for e and t; (2)–(3) scenarios in computing the cost $c(d_{\min}(e \uparrow t))$; (4)–(7) scenarios in computing the cost $c(d_{\min}(e \downarrow t))$. Case names: (1) Basis. (2) Duplication with directions as in Fig. 4(2). (3) Transfer into the tube t_e; the case of a subsequent speciation is illustrated, while the analogous case of a subsequent arrival at a leaf is omitted. For a non-leaf edge e, we define $t_e = \arg\min\{\min_{t'}\{c(d_{\min}(e_1 \downarrow t'_1)) + c(d_{\min}(e_2 \downarrow t'_2)), c(d_{\min}(e_1 \downarrow t'_2)) + c(d_{\min}(e_2 \downarrow t'_1))\}\}$. For a leaf edge e, we define t_e as a leaf tube sharing the same label as e, which yields $c(d_{\min}(e \uparrow t)) = 1$. (4) Non-transfer-associated speciation. (5) Transfer-associated speciation. (6) Loss. (7) Duplication with directions as in Fig. 4(7).

The embedding F is thus constructed, and the description of the algorithm continues below.

2. The following lemma establishes the first critical property of the embedding F.

Lemma 2. *The embedding F is dead-end-free.*

The proof of this lemma straightforwardly follows from the description of the algorithm and is omitted here.

3. The following lemma establishes the second critical property of the embedding F.

Lemma 3. *Suppose that in a dead-end-free embedding f, a conflict exists between edges e_1 and e_2 at a vertex t_-, and there are no conflicts at any descendant vertex of t_-. Then only the edges e_1 and e_2 exit from t, and each of them undergoes either a loss or a transfer-associated speciation at t_-.*

The proof of this lemma straightforwardly follows from the fact that the edges e_1 and e_2 are not dead ends, and is omitted here.

Second stage of the algorithm. From the embedding $F\colon T_1 \to T_2$, we construct an embedding $F'\colon T_1 \to T_2$ (denoted by $F \le F'$) that resolves a single conflict in F without introducing new ones, while adding exactly one new transfer. By doing so, we iteratively transform the initial embedding F into a sequence $F \le F' \le \ldots \le F^{\#}$ until a conflict-free embedding $F^{\#}\colon T_1 \to T_2$ is obtained.

Below, we prove that the number of added transfers in this process is bounded by the sum of the number of losses and transfer-associated speciations in the initial embedding F.

Third stage of the algorithm. From $F^{\#}$, we construct the final collection $S'(F^{\#})$ of clade pairs as described in Lemma 1. Specifically, for each transfer of an edge e under $F^{\#}$, we include the pair (K_e, K_t) in $S'(F^{\#})$, where the tube t is the destination of the transfer of e. Simultaneously, we determine the cardinality $|S'(F^{\#})|$. The algorithm outputs the partial bijection $S'(F^{\#})$ along with its cardinality $|S'(F^{\#})|$, thereby providing an approximate value for the distance $R(T_1, T_2)$.

Let us describe one such iteration, specifically the transition from F to F'. Suppose that in a dead-end-free embedding F, a conflict exists between edges e_1 and e_2 at a vertex $t_- \in T_2$, such that no conflicts occur below t_- in T_2. By Lemma 3, only e_1 and e_2 exit from t. Consider the tube t' where the edge e_1 originates—that is, the tube containing (or at the root of which lies) $F(e_{1+})$. If e_1 undergoes a loss at t_-, we transfer e_1 from t' to the descendant tube of t containing (or at the end of which lies) $F(e_{1-})$ (see the first transformation in Fig. 5(1)). If e_1 undergoes a transfer-associated speciation at t_-, then one of its child edges, say e_1', enters the child tube t_1 of t from above, and either e_1' itself or all of its duplicated descendants are transferred from t_1. The other child edge e_1'' enters t_2. We duplicate the edge e_1 within t' and transfer e_1'' to the descendant tube of t containing (or at the end of which lies) $F(e_{1-}'')$. Furthermore, if e_1' is a pre-transfer edge, we duplicate it multiple times again and transfer these duplicated descendants into the same tubes to which they were transferred under F (Fig. 5(2)).

By doing so, the conflict at t_- is resolved. Since a new transfer (either e_1 or e_1') has been introduced, new dead-end edges may have emerged above it in T_1.

Lemma 4. *The newly emerged dead-end edges form a contiguous segment in T_1 located on the path upward from the new transfer, possibly separated from it by a segment of pre-transfer edges.*

The statement of this lemma straightforwardly follows from the construction, and the proof is omitted here.

Let us consider the highest edge $e^\star$ within the segment of dead-end edges and eliminate it by reconstructing the embedding so that $e^\star$ becomes a loss-free pre-transfer edge, thereby ceasing to be a dead end. To achieve this, we duplicate $e^\star$ multiple times within the tube $t^\star$ where it originates, until all of its duplicated descendants belong to the separating set $D_{e^\star}$. We then transfer them from $t^\star$ directly into the tubes that serve as the destinations of the transfers for the edges in $D_{e^\star}$ (see the second transformation in Fig. 5(1) and the separate illustration of dead-end edge elimination in Fig. 6).

It is straightforward to see that this reconstruction eliminates all dead-end edges (the details are omitted here). We then proceed to the next iteration.

Thus, we obtain a *dead-end-free* and *conflict-free embedding* $F^{\#}\colon T_1 \to T_2$, whose number of transfers (as proven below) increases by at most $B(F) + C(F)$. From $F^{\#}$, we construct the final collection $S'(F^{\#})$ of clade pairs as described in Lemma 1 (in the third stage of the algorithm). Specifically, for each transfer of an edge e under $F^{\#}$, we include the pair (K_e, K_t) in $S'(F^{\#})$, where the tube t is the destination of the transfer of e. The algorithm outputs the partial bijection $S'(F^{\#})$,

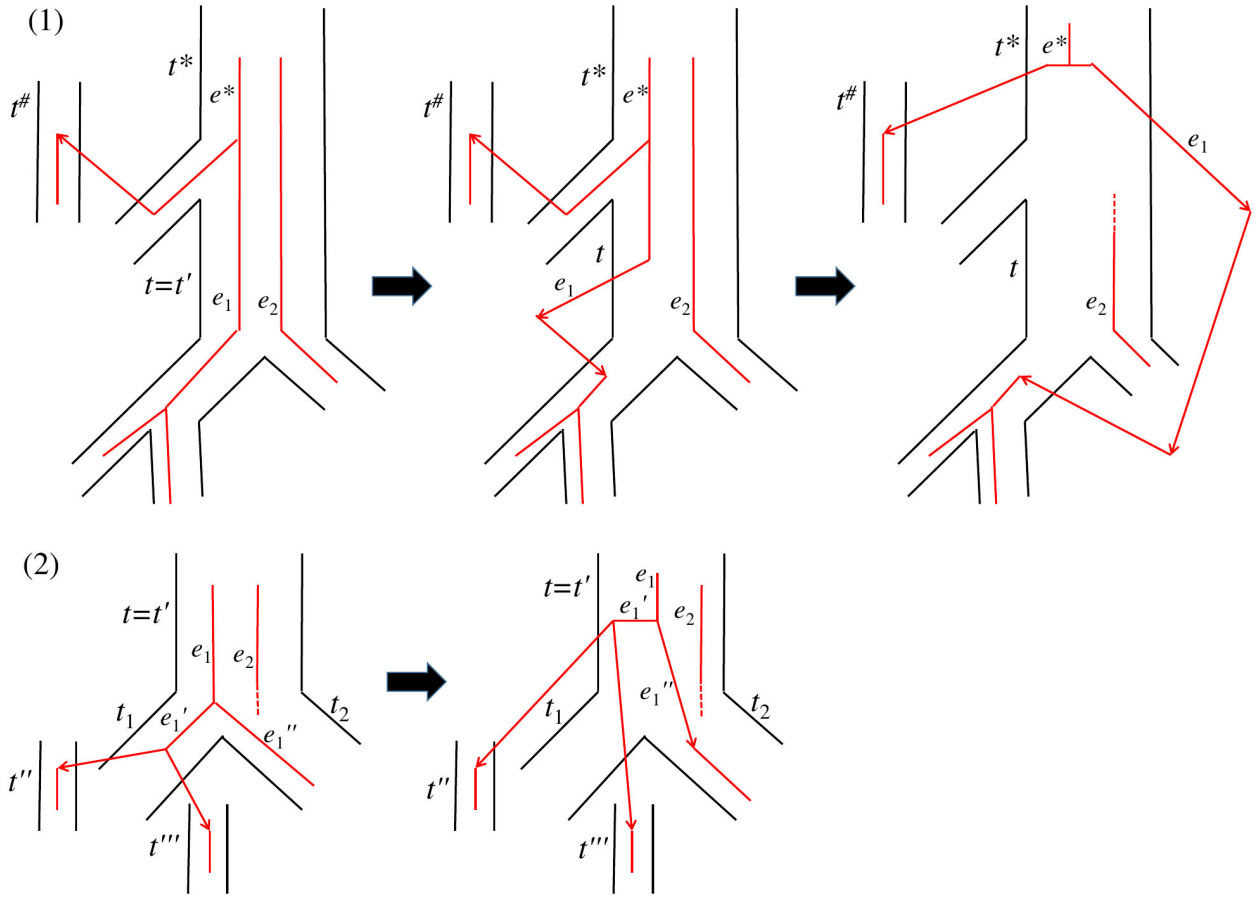


Fig. 5. (1) Left: the embedding $F\colon T_1 \to T_2$. Middle: the embedding $F'\colon T_1 \to T_2$. The transformation of F into F' resolves the conflict between edges e_1 and e_2, followed by the elimination of the edge $e^\star$ that became a dead end. The scenario illustrates a loss of e_1 in the special case where $t = t'$. Right: the embedding after this elimination is depicted. (2) Left: the embedding $F\colon T_1 \to T_2$. Right: the embedding $F'\colon T_1 \to T_2$, which resolves the conflict between edges e_1 and e_2 in the case where e_1 undergoes a speciation under the same special case. Here, e_1' acts as a pre-transfer edge with two duplicated descendants.

providing an approximate distance $R(T_1, T_2)$ (the proof of the error bound is presented in Section 5). This concludes the description of the algorithm.

5. ERROR BOUNDS AND COMPLEXITY ESTIMATION

Theorem 1. *The algorithm described above computes an approximation of the distance $R(T_1, T_2)$ for the trees T_1 and T_2, and constructs a corresponding consistent partial bijection $S'(F^{\#})$ such that its cardinality minus one exceeds the exact value by a multiplicative factor of at most two. The running time of the algorithm is quadratic in the number of leaves.*

Proof sketch. For any formula $d(e \downarrow t)$ or $d(e \uparrow t)$, the formulas $d'(e_1 \downarrow t_1), d'(e_1 \uparrow t_1), \ldots,$ appearing on the right-hand sides of the minimizing sums in Fig. 4, are called *child* formulas. Let $F(e \downarrow t)$ and $F(e \uparrow t)$ denote the embeddings of the subtree T_e into T_2 where the superroot of T_e is mapped to the tube t rather than to the external tube. These embeddings are constructed from the cost $c(\cdot)$ during the backward pass by following the pointers, starting from the formulas $d_{\min}(e \downarrow t)$

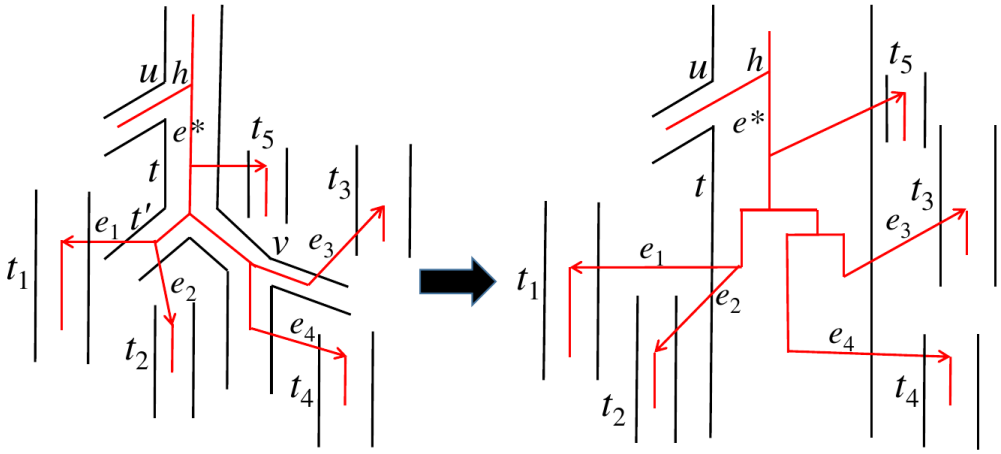


Fig. 6. The reconstruction of an embedding $f\colon T_1 \to T_2$ (left) into an embedding $F\colon T_1 \to T_2$ (right) that eliminates the dead-end property of the edge $e^\star$. Here, the tube t is the tube where $e^\star$ originates.

and $d_{\min}(e \uparrow t)$, respectively. The restrictions of $F(e \downarrow t)$ and $F(e \uparrow t)$ that originate from child formulas are called *child embeddings*.

It is straightforward to verify that the cost of the embedding $F(e \downarrow t)$ or $F(e \uparrow t)$ equals the sum of the cost of the event occurring at the pair (e, t) (i.e., of the edge e within the tube t) and the costs of the child embeddings induced by this event.

The subsequent lemma implies that in any embedding f, one can eliminate a transfer-associated speciation of an edge e if both of its child edges undergo a transfer or act as a pre-transfer edge (consequently making e a dead end), such that neither the cost of the resulting embedding F nor the number of its transfers increases.

Lemma 5. *The reconstruction of an embedding f illustrated in Fig.* 6, *which eliminates a dead-end edge $e^\star$, does not increase the cost of the resulting embedding F, preserves the conflict-free property, and does not increase the number of transfers.*

The proof straightforwardly follows from the construction in Fig. 6 and is omitted here.

For any embedding, a transfer or a pre-transfer of a root edge e within a tube t (where the superroot is mapped to t) is denoted by $\Uparrow$, whereas the alternative condition is denoted by $\Downarrow$. (We show below that $\Uparrow$ corresponds to $\uparrow$ and $\Downarrow$ corresponds to $\downarrow$, meaning that the costs $c(d_{\min}(e \uparrow t))$ and $c(d_{\min}(e \downarrow t))$ computed during the forward pass of the algorithm are precisely equal to the costs of the minimal embeddings $T_e \to T_2$ satisfying $\Uparrow$ and $\Downarrow$, respectively.)

We define an $\Uparrow$-descendant and a $\Downarrow$-descendant as a descendant that is either transferred or acts as a pre-transfer edge, and a descendant that does not satisfy this condition, respectively. For a pair (e, t)—that is, an edge e within a tube t—the following event types 1–6 can occur, as listed at the beginning of Section 2 (for brevity, we refer to them simply as "events"). Events 7–12 can be replaced by events 1–6 without increasing the embedding cost or the number of transfers (for event 7, this is proven in Lemma 5; for the remaining cases, the reasoning is analogous, straightforwardly follows from Figs. 7(1)–7(4), and is omitted here).

1. Non-transfer-associated speciation (Fig. 4(4)).
2. Transfer-associated speciation with exactly one $\Downarrow$-descendant (Fig. 4(5)).
3. Loss of an edge that remains a $\Downarrow$-edge (Fig. 4(6)).
4. Duplication of a pre-transfer edge (Fig. 4(2)).
5. Duplication of a non-pre-transfer edge (Fig. 4(7)).

6. Transfer followed by either a non-transfer-associated speciation or arrival at a basis pair (Fig. 4(3)).
7. Transfer-associated speciation followed by two $\Uparrow$-descendants; this is replaced as described in Lemma 5 (Fig. 6).
8. Loss of an edge that becomes an $\Uparrow$-edge; this is replaced analogously to event 7 (Fig. 7(1)).
9. Transfer followed by a loss; this is replaced analogously (Fig. 7(2)).
10. Transfer followed by a duplication; this is replaced by a sequence of duplications followed by transfers (Fig. 7(3)).
11. Transfer followed by a transfer-associated speciation; this is replaced analogously (Fig. 7(4)).
12. Transfer followed by another transfer; this is eliminated by merging the two transfers into a single one.

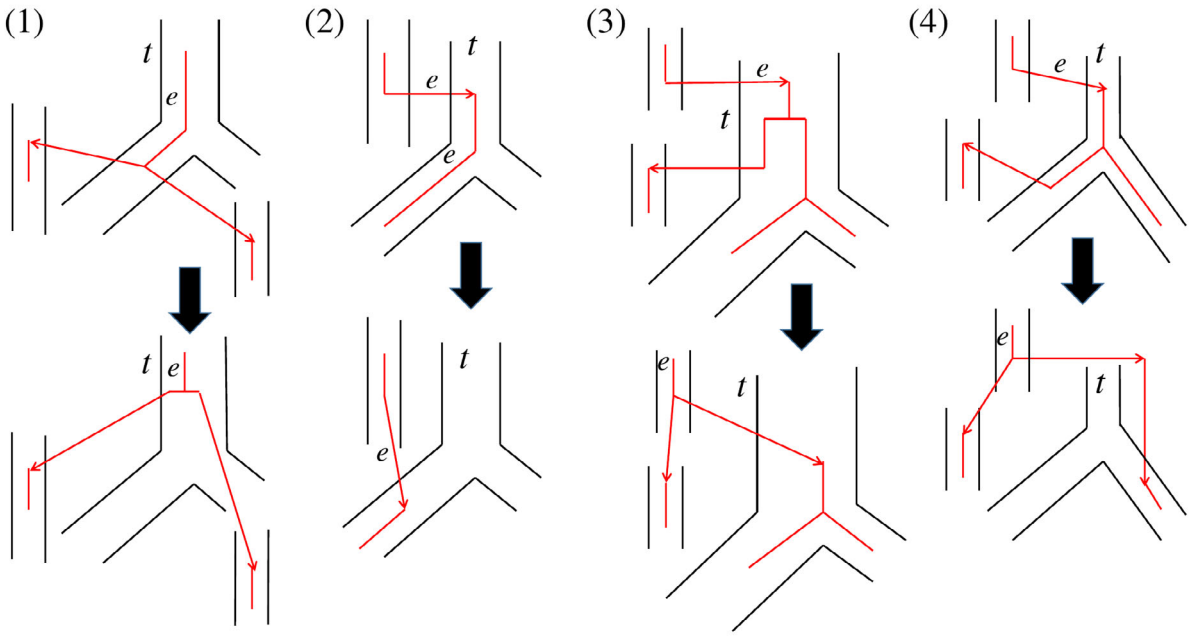


Fig. 7. (1) Elimination of a loss followed by a transfer of duplicated descendants. (2) Elimination of a transfer followed by a loss. (3) Elimination of a transfer followed by a duplication. (4) Elimination of a transfer followed by a transfer-associated speciation.

Duplication with two $\Downarrow$-descendants is not allowed by definition.

The following two lemmas establish the correctness of the forward pass of dynamic programming in the first stage of the algorithm. Their proofs are straightforward and are omitted here.

Lemma 6. *For each pair (e,t), the costs of the embeddings $F(e\downarrow t)$ and $F(e\uparrow t)$ are precisely equal to the costs $c(d_{\min}(e\downarrow t))$ and $c(d_{\min}(e\uparrow t))$, respectively.*

Let $(\star)$ denote the class of embeddings that feature only events 1–6.

Lemma 7. *For any non-basis pair (e,t), the cost satisfies*

$$c(d_{\min}(e\uparrow t)) = \min\{c(f) \mid f\colon T_e \to T_2 \text{ satisfies } \Uparrow \text{ under } (\star)\},$$

and for an any pair (e,t), the cost satisfies

$$c(d_{\min}(e\downarrow t)) = \min\{c(f) \mid f\colon T_e \to T_2 \text{ satisfies } \Downarrow \text{ under } (\star)\}.$$

We now continue the proof of Theorem 1. A loss or a transfer-associated speciation event occurring at a vertex where a conflict exists is called *conflict-associated*.

Lemma 8. *Let the iterations originate from the embedding F_0 obtained in the first stage of the algorithm, where only events* 1–6 *are permitted. Each iteration transforming the current embedding F (ultimately yielding $F^{\#}$) resolves a conflict at t_-, introduces no new conflicts, and strictly decreases the total number of conflict-associated events.*

Proof sketch. Consider a single iteration of the transformation of F_0 into $F^{\#}$ (Fig. 5). Let F denote the embedding prior to this iteration. At the vertex t_- where the iteration resolves the conflict, two conflict-associated events are eliminated. We now examine where new conflicts or conflict-associated events might emerge. It is straightforward to see that for any pair consisting of an edge $e \in T_1$ and a vertex $v \in T_2$, the iteration either preserves the event of a loss or a speciation of e at v, or removes it altogether; no new losses or speciations of e at v are introduced. Consequently, no new conflicts arise either.

Note that the absence of new conflicts does not immediately imply that no new conflict-associated events appear. Although no new pairs of edges exiting a tube are introduced, a new conflict-associated event can emerge at a conflicting vertex u if and only if a non-transfer-associated speciation at u becomes transfer-associated.

We now address the second statement of the lemma. The replacement of a non-transfer-associated speciation with a transfer-associated one can occur only when $u = t'_+$, since t' is the unique tube where new images of vertices from T_1 appear. Two scenarios are possible. In the first scenario, the ancestor e' of the edge e_1, for which $F(e'_-) = t'_+$, becomes a dead end ($e' = e^\star$ in Fig. 5(1)); after the elimination of dead-end edges, the transfer-associated speciation at t'_+ disappears, while a new transfer-associated speciation can emerge only at $F(e^\star_+)$, where $e^\star$ is the highest edge that became a dead end. In the second scenario, the edge e' does not become a dead end. In this case, no dead-end edges arise, the iteration consists solely of the first stage, and a new transfer-associated speciation can appear only at $t'_+ = F(e_{1+})$ (this scenario would occur if, in Fig. 5(1), the child edge of $e^\star$ were not transferred to $t^{\#}$ but instead had a transfer-free continuation to a leaf).

Thus, each iteration eliminates two conflict-associated events while generating at most one new event. Consequently, the total number of conflict-associated events strictly decreases at each iteration. $\triangle$

Lemma 9. *In any conflict-free embedding $f\colon T_1 \to T_2$, the total number of losses $B(f)$ and transfer-associated speciations $C(f)$ is strictly less than the number of transfers $A(f)$.*

Proof. We construct a function g that maps each loss and transfer-associated speciation to a unique transfer, and show that g is injective but not surjective. We fix an arbitrary function $\ell(t)$ that associates each tube t with a designated leaf ℓ located below it. Suppose that an edge e undergoes a loss at t_-. Consider a path p originating from t_- and passing through the child tube t' of t (which e does not enter) to the leaf $\ell(t')$. Since a leaf edge from T_1 must reach this leaf, the first tube encountered along p containing an incoming edge will be the destination of some transfer. We map this transfer to the loss under consideration. Analogously, suppose that e terminates with a transfer-associated speciation at t_-. Consider a path p originating from t_- and passing through the child tube t' of t from which the child edge of e (or all of its duplicated descendants) is transferred, to the leaf $\ell(t')$. As in the case of a loss, the first tube encountered along p will be the destination of a transfer. We map this transfer to the transfer-associated speciation under consideration.

Let us prove the injectivity of the function g. Prior to reaching the destination tube of the transfer, the path p passes exclusively through *free vertices* in T_2 (excluding its tail), i.e., vertices that contain neither f-images of vertices from T_1 nor losses of any edges from T_1. Consider a transfer

of an edge e. We trace a path upward in T_2, starting from its destination tube, until the first non-free vertex u is encountered. At u, there must exist either a loss or a transfer-associated speciation; otherwise, a non-free vertex would have been encountered earlier. The transfer of e can only be mapped to this specific loss or transfer-associated speciation. Indeed, by virtue of the conflict-free property, u harbors no other event that could serve as a g-preimage of e, and no such event exists above u because, as noted above, the path p passes only through free vertices. This establishes the injectivity of g. Finally, consider a transfer into a tube above which no other destination tubes of any transfers exist (any valid embedding contains at least one transfer, e.g., the transfer from the external tube). This transfer lacks a g-preimage because a transfer destination tube must exist above any non-free vertex. Consequently, g is not surjective, which completes the proof. $\triangle$

Conclusion of the proof of Theorem 1. Let the embedding F (prior to the resolution of conflicts) feature $A(F)$ transfers, $B(F)$ losses, and $C(F)$ transfer-associated speciations. Thus, F contains at most $B(F)+C(F)$ conflict-associated events. During the transformation of F into $F^{\#}$, each iteration adds exactly one transfer and, by Lemma 8, strictly decreases the number of conflict-associated events. Consequently, at most $B(F)+C(F)$ transfers are added in total. Now, consider a conflict-free embedding $F^{\star}\colon T_1 \to T_2$ that minimizes the number of transfers $A(F^{\star})$ among all conflict-free embeddings, and let it feature $B(F^{\star})$ losses and $C(F^{\star})$ transfer-associated speciations. Recall that the cost $c(F)$ is minimal not only among all embeddings restricted to events 1–6, but also among those featuring any events 1–12, owing to the elimination rules for events 7–12. From this, we obtain:

$$A(F^{\#}) \le c(F) \le c(F^{\star}) = A(F^{\star}) + \big(B(F^{\star}) + C(F^{\star})\big) < 2A(F^{\star}),$$

where the first inequality follows from the number of iterations, the second from the minimality of $c(F)$, and the third from Lemma 9. Since these values are integers, we obtain

$$A(F^{\#}) \le 2A(F^{\star}) - 1,$$

which implies

$$|S'(F^{\#})| - 1 \le 2(|S^{\star}| - 1),$$

thereby completing the proof of the upper bound.

At the final stage, the algorithm constructs a consistent partial bijection $S'(F^{\#})$ from $F^{\#}$ according to Lemma 1, such that its cardinality equals the number of transfers $A(F^{\#})$. By the second statement of Lemma 1, if there existed a consistent partial bijection whose cardinality minus one were strictly less than $(A(F^{\#}) - 1)/2$, there would exist a conflict-free embedding $T_1 \to T_2$ with fewer transfers than $A(F^{\star})$. This directly contradicts the minimality of $A(F^{\star})$ and establishes the multiplicative error bound of the algorithm.

The most computationally intensive part of constructing the embedding is evaluating the formulas $d(e \uparrow t)$ and $d(e \downarrow t)$ during the forward pass. The time complexity of this stage is $\mathcal{O}(|T_1||T_2|)$, where $|T|$ denotes the number of leaves in T. Indeed, the number of evaluated formulas is $\mathcal{O}(|T_1||T_2|)$. For each edge e, the tube t_e is determined first, which requires $\mathcal{O}(|T_2|)$ time; subsequently, at most ten cases are examined for each pair (e, t). The conflict resolution stage is also executed in quadratic time. Specifically, Lemma 8 implies that it suffices to iterate over the vertices of T_2 exactly once, proceeding from the leaves to the root. Therefore, the process requires at most $|T_2|$ iterations. Each iteration involves redefining the images of the vertices from T_1 and is executed in time linear with respect to $|T_1|$. $\triangle$

6. CONCLUSIONS

We have presented a quadratic-time algorithm for computing the MAF distance between two rooted binary trees without any structural restrictions, and have proven a multiplicative approxi-

mation factor of 2. We have also described a model for the evolution of one tree relative to another, conceptualized as discrete time steps. Future research could focus on improving this approximation bound (although an arbitrarily accurate approximation scheme cannot be expected due to the APX-hardness of the problem [13]), as well as extending the method to non-binary or unrooted trees. Furthermore, it is of significant interest to generalize our algorithm to the case of multiple trees. Existing algorithms for multiple trees are described in [5], achieving a multiplicative error bound of 3 for rooted trees and 4 for unrooted trees. Additionally, Reference [6] provides a simpler 4-approximation for unrooted trees, and [14] investigates the multiple-tree case, proving that the corresponding maximum agreement forest problem is MAXSNP-complete.

FUNDING

This research was supported by the Russian Science Foundation under grant no. 24-44-00099 (https://rscf.ru/project/24-44-00099/).

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. Gorbunov, K. and Lyubetsky, V., Linear Time Additively Exact Algorithm for Transformation of Chain-Cycle Graphs for Arbitrary Costs of Deletions and Insertions, *Mathematics*, 2020, vol. 8, no. 11, p. 2001 (30 pp.). https://doi.org/10.3390/math8112001
2. Gorbunov, K.Yu. and Lyubetsky, V.A., An Exact Quadratic Algorithm for the Shortest Tree Transformation, *Dokl. Ross. Akad. Nauk*, 2024, vol. 519, no. 1, pp. 22–27 [*Dokl. Math.* (Engl. Transl.), 2024, vol. 110, no. 2, pp. 373–378]. https://doi.org/10.1134/S1064562424702259
3. Gorbunov, K.Yu. and Lyubetsky, V.A., On an Algorithm for Reconciling Gene and Species Trees Taking into Account Duplications, Losses, and Horizontal Gene Transfers, *Inform. Protsessy*, 2010, vol. 10, no. 2, pp. 140–144. http://www.jip.ru/2010/140-144-2010.pdf
4. Olver, N., Schalekamp, F., van der Ster, S., Stougie, L., and van Zuylen, A., A Duality-Based 2-Approximation Algorithm for Maximum Agreement Forest, *Math. Program.*, 2023, vol. 198, no. 1, pp. 811–853. https://doi.org/10.1007/s10107-022-01790-y
5. Chen, J., Shi, F., and Wang, J., Approximating Maximum Agreement Forest on Multiple Binary Trees, *Algorithmica*, 2016, vol. 76, no. 4, pp. 867–889. https://doi.org/10.1007/s00453-015-0087-6
6. Dempsey, J., van Iersel, L., Jones, M., and Zeh, N., A Simple 4-Approximation Algorithm for Maximum Agreement Forests on Multiple Unrooted Binary Trees, *Inf. Process. Lett.*, 2025, vol. 190, p. 106572 (5 pp.). https://doi.org/10.1016/j.ipl.2025.106572
7. Hein, J., Jiang, T., Wang, L., and Zhang, K., On the Complexity of Comparing Evolutionary Trees, *Discrete Appl. Math.*, 1996, vol. 71, no. 1–3, pp. 153–169. https://doi.org/10.1016/S0166-218X(96)00062-5
8. Bordewich, M. and Semple, C., On the Computational Complexity of the Rooted Subtree Prune and Regraft Distance, *Ann. Comb.*, 2004, vol. 8, no. 4, pp. 409–423. https://doi.org/10.1007/s00026-004-0229-z
9. Whidden, C., and Matsen, F.A., Calculating the Unrooted Subtree Prune-and-Regraft Distance, *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2019, vol. 16, no. 3, pp. 898–911. https://doi.org/10.1109/TCBB.2018.2802911
10. Schalekamp, F., van Zuylen, A., and van der Ster, S., A Duality-Based 2-Approximation Algorithm for Maximum Agreement Forest, in *Proc. 43rd Int. Colloq. on Automata, Languages, and Programming (ICALP 2016)*, Leibniz Int. Proc. Inform. (LIPIcs), vol. 55, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016, pp. 70:1–70:14. https://doi.org/10.4230/LIPIcs.ICALP.2016.70

11. Chen, Z.-Z., Harada, Y., and Wang, L., A New 2-Approximation Algorithm for rSPR Distance, *Proc. Int. Conf. on Bioinformatics Research and Applications (ICBRA 2017)*, Cai, Z., Daescu, O., and Li, M., Eds., Cham: Springer, 2017, Lect. Notes Bioinform., vol. 10330, pp. 128–139. https://doi.org/10.1007/978-3-319-59575-7_12

12. Gorbunov, K.Yu. and Lyubetsky, V.A., Distance between Binary Trees and Evolution of One Tree Relative to Another, *Probl. Peredachi Inf.*, 2026, vol. 62, no. 3 (to appear).

13. Rodrigues, E.M., Algoritmos para comparação de árvores filogenéticas e o problema dos pontos de recombinação, *PhD Thesis*, Univ. of São Paulo, Brazil, 2003, Chapter 7. https://teses.usp.br/teses/disponiveis/45/45132/tde-20210729-131755/publico/RodriguesEstelaMaris.pdf

14. Chataigner, F., Approximating the Maximum Agreement Forest on k Trees, *Inf. Process. Lett.*, 2005, vol. 93, no. 5, pp. 239–244. https://doi.org/10.1016/j.ipl.2004.11.004

Publisher's Note. Pleiades Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

AI tools may have been used in the translation or editing of this article.