Algorithmica (2019) 81:3692–3706
https://doi.org/10.1007/s00453-019-00594-5

An $O(n \log n)$ Time Algorithm for Computing the Path-Length Distance Between Trees

David Bryant[1] · **Celine Scornavacca**[2]

Received: 10 October 2018 / Accepted: 28 May 2019 / Published online: 19 June 2019

Abstract
Tree comparison metrics have proven to be an invaluable aide in the reconstruction and analysis of phylogenetic (evolutionary) trees. The path-length distance between trees is a particularly attractive measure as it reflects differences in tree shape as well as differences between branch lengths. The distance equals the sum, over all pairs of taxa, of the squared differences between the lengths of the unique path connecting them in each tree. We describe an $O(n \log n)$ time for computing this distance, making extensive use of tree decomposition techniques introduced by Brodal et al. (Algorithmica 38(2):377–395, 2004).

Keywords Phylogeny · Tree comparison metrics · Path-length metric · Tree decomposition

Mathematics Subject Classification 68Q25 · 92D15 · 05CO5

1 Introduction

A *phylogenetic tree* is a tree describing the evolution of a set of entities X (species, genes etc.), which will be called *taxa* from now onwards. Here trees are considered as undirected, or *unrooted*. Degree-one nodes are called *leaves* and a bijective function associates each taxon to a leaf. Internal nodes represent putative ancestral taxa and branch lengths quantify the evolutionary distances between nodes.

Tree comparison metrics provide a quantitative measure of the similarity or difference between two phylogenetic trees. They have proven invaluable for statistical

✉ David Bryant
david.bryant@otago.ac.nz

Celine Scornavacca
celine.scornavacca@umontpellier.fr

1 Department of Mathematics and Statistics, University of Otago, Dunedin, New Zealand

2 ISEM, CNRS, Université de Montpellier, IRD, EPHE, Montpellier, France

testing (e.g. [9,11,13]), for visualisation [8], and for the construction of consensus trees [3,10,14]. By far the most well-known tree comparison metric is the Robinson–Foulds metric [12], which equals the number of bipartitions[1] that are in one tree and not the other. However many other different metrics have also been proposed, each one based on a different characteristic of the trees being compared.

Here we consider pairs of trees on the same set of taxa. Also, our trees are *binary*, i.e. each internal node has degree three. The *path-length* between two taxa in a phylogenetic tree is the sum of the branch lengths along the unique path between them. The *path-length distance* between two trees T_1 and T_2 is given by

$$\Delta(T_1, T_2) = \sum_{ij} (p_{ij} - q_{ij})^2, \tag{1}$$

where p_{ij} is the path length between taxa i and j in the first tree and q_{ij} is the path length in the second tree. We note that $\sqrt{\Delta(T_1, T_2)}$ is a metric in the mathematical sense. The first explicit description of the metric appears in [11] (without branch lengths) and [10] (with branch lengths), though closely related ideas appear much earlier (e.g. [6,7,15]).

Given a phylogeny with n leaves, it takes $O(n^2)$ time to construct the set of all path-lengths $p_{12}, p_{13}, \ldots, p_{(n-1)n}$, using the dynamic programming algorithm presented in [4]. Hence the path-length distance can be easily computed in $O(n^2)$ time. Our main contribution in this paper is to show that we can compute this distance in $O(n \log n)$ time, which is almost, but not quite, linear in the size of the problem input.

Expanding (1) gives

$$\Delta(T_1, T_2) = \sum_{ij} (p_{ij})^2 + \sum_{ij} (q_{ij})^2 - 2 \sum_{ij} p_{ij} q_{ij}. \tag{2}$$

The first two terms can be evaluated in linear time using dynamic programming, as outlined in Sect. 2. To compute the third term efficiently we first introduce a tree decomposition technique (Sect. 3) allowing the sum to be evaluated in $O(n \log n)$ time (Sect. 4). Both the tree decomposition and algorithm of Sect. 4 draw heavily on an algorithm of [2] for computing the quartet distance between two trees.

2 Sums of Squared Distances

In this section we show how to compute the sum of squared distances $\sum_{ij} p_{ij}^2$ in a tree in linear time. We begin by introducing some notation that will be used in the rest of the paper.

Select an arbitrary leaf ρ and consider both T_1 and T_2 as rooted trees with root ρ. We think of ρ being at the top of the tree and the other leaves being at the bottom of the tree. For any two edges e, e' we write $e \preceq e'$ if the path from e to ρ passes through

[1] A bipartition $A|B$ with $A \cup B = X$ is in a phylogenetic tree $T = (V, E)$ if there exists an edge $e \in E$ such that its removal creates two trees with taxon sets A and B.

e'. We write $e \prec e'$ if $e \preceq e'$ and $e \neq e'$. Hence if e is the edge incident with the root ρ then $e' \prec e$ for all other edges e'. We say that e is external if it is incident to a leaf other than ρ; otherwise e is internal. When e is internal let e_L and e_R denote the edges incident and immediately below e.

We will use e, e' to denote edges in T_1 and f, f' to denote edges in T_2. We let x_e denote the length of an edge e in T_1 and y_f the length of an edge f in T_2. Let A_{ij} denote the set of edges on the path from i to j in T_1 and let B_{ij} denote the corresponding set in T_2. Hence

$$p_{ij} = \sum_{e \in A_{ij}} x_e \qquad q_{ij} = \sum_{f \in B_{ij}} y_f.$$

Let $n(e)$ denote the number of leaves ℓ such that the path from ℓ to ρ passes through e. Define

$$\alpha(e) = \sum_{e' \preceq e} n(e') x_{e'}.$$

Proposition 1

$$\sum_{ij} (p_{ij})^2 = \sum_{e \text{ internal}} \Big[x_e (n - n(e))(2\alpha(e) - n(e)x_e) + 2\alpha(e_L)\alpha(e_R) \Big] + \sum_{e \text{ external}} \Big[x_e^2 (n - n(e)) n(e) \Big].$$

Proof Given two edges e_1, e_2 we let

$$\chi(e_1, e_2) = |\{\text{pairs } ij : e_1, e_2 \in A_{ij}\}|, \tag{3}$$

the number of pairs of taxa having both e_1 and e_2 on the path between them. Then

$$\begin{aligned} \sum_{ij} (p_{ij})^2 &= \sum_{ij} \left(\sum_{e_1 \in A_{ij}} x_{e_1} \right) \left(\sum_{e_2 \in A_{ij}} x_{e_2} \right) \\ &= \sum_{ij} \sum_{e_1, e_2 \in A_{ij}} x_{e_1} x_{e_2} \qquad (4) \\ &= \sum_{e_1} \sum_{e_2} \sum_{ij : e_1, e_2 \in A_{ij}} x_{e_1} x_{e_2} \qquad (5) \\ &= \sum_{e_1} \sum_{e_2} x_{e_1} x_{e_2} \chi(e_1, e_2) \qquad (6) \end{aligned}$$

If $e_1 \prec e_2$ then $e_1 \neq e_2$ and $\chi(e_1, e_2) = n(e_1)(n - n(e_2))$. Hence, for any e_2 we have

$$\begin{aligned}\sum_{e_1:e_1\prec e_2} x_{e_1}x_{e_2}\chi(e_1,e_2) &= x_{e_2}(n-n(e_2)) \sum_{e_1:e_1\prec e_2} x_{e_1}n(e_1)\\ &= x_{e_2}(n-n(e_2))(\alpha(e_2)-n(e_2)x_{e_2}).\end{aligned}$$

If $e_1 \not\preceq e_2$ and $e_2 \not\preceq e_1$ then $\chi(e_1, e_2) = n(e_1)n(e_2)$. Furthermore there is an edge e with children e_L, e_R such that, without loss of generality, $e_1 \preceq e_L$ and $e_2 \preceq e_R$. For such an edge e we have

$$\begin{aligned}\sum_{e_1:e_1\preceq e_L}\sum_{e_2:e_2\preceq e_R} x_{e_1}x_{e_2}\chi(e_1,e_2) &= \sum_{e_1:e_1\preceq e_L}\sum_{e_2:e_2\preceq e_R} x_{e_1}x_{e_2}n(e_1)n(e_2)\\ &= \alpha(e_L)\alpha(e_R).\end{aligned}$$

Summing up over all e_1, e_2 in (6) we have

$$\begin{aligned}\sum_{ij}(p_{ij})^2 &= \sum_{e_1}\sum_{e_2} x_{e_1}x_{e_2}\chi(e_1,e_2)\\ &= 2\sum_{e_2}\sum_{e_1\prec e_2} x_{e_1}x_{e_2}\chi(e_1,e_2) + 2\sum_{e}\sum_{e_1\preceq e_L}\sum_{e_2\preceq e_R} x_{e_1}x_{e_2}\chi(e_1,e_2)\\ &\quad + \sum_e x_e x_e \chi(e,e)\\ &= 2\sum_{e_2 \text{ internal}} x_{e_2}(n-n(e_2))(\alpha(e_2)-n(e_2)x_{e_2}) + 2\sum_{e \text{ internal}} \alpha(e_L)\alpha(e_R)\\ &\quad + \sum_e x_e x_e n(e)(n-n(e))\end{aligned}$$

and the result follows. □

Proposition 2 *The sum $\sum_{ij}(p_{ij})^2$ can be computed in linear time.*

Proof If e is external, $n(e) = 1$ and $\alpha(e) = x_e$. Otherwise

$$n(e) = n(e_L) + n(e_R)$$

and

$$\begin{aligned}\alpha(e) &= \sum_{e'\preceq e} n(e')x_{e'}\\ &= \sum_{e'\preceq e_L} n(e')x_{e'} + \sum_{e'\preceq e_R} n(e')x_{e'} + n(e)x_e\\ &= \alpha(e_L) + \alpha(e_R) + n(e)x_e.\end{aligned}$$

Hence with a post-order traversal of the tree we can compute $n(e)$ and $\alpha(e)$ for all edges e in $O(n)$ time. Computing the sum takes a further $O(n)$ time by Proposition 1. $\sum_{ij}(q_{ij})^2$ can be computed in the same way. □

3 Segment Decomposition

In this section we introduce a hierarchical decomposition of the edge set of T_2 that forms the structure used in our dynamical programming algorithm in Sect. 4.

Let Q be a connected subset of $E(T_2)$, the set of edges of T_2. We define the *boundary* of Q to be the set of vertices incident both to edges within Q and to edges outside Q:

$$\partial Q = \{v : \text{ there are } e \in Q, e' \notin Q \text{ incident with } v\}.$$

The *degree* of Q is the cardinality of ∂Q. A *segment* of T_2 is a connected subset of $E(T_2)$ with degree at most two.

A *segment decomposition* for T_2 is a binary tree T_D such that

(D1) The leaves of T_D correspond to edges in $E(T_2)$ (i.e. minimal segments);
(D2) Each node of T_D corresponds to a segment of T_2;
(D3) The segment corresponding to an internal node of T_D equals the disjoint union of the segments corresponding to its children.

An example of segment decomposition is given in Fig. 1.

The main result in this section is that we can, in linear time, construct a segment decomposition for T with height $O(\log n)$.

The definition of a segment decomposition is based on the tree decomposition used by [2] to compute quartet-based distances, which in turn are based on techniques for efficient parsing of expressions [1,5]. The main difference with [2] is that the segment decomposition is based on partitioning the set of edges, rather that the set of vertices, and that we were able to obtain a tighter bound on the height.

Our algorithm for constructing T_D is agglomerative. At each stage, the nodes in T_D form a forest, each component being a rooted tree with a unique maximal (root) node. We start with a single degree one vertex for each edge in $E(T_2)$; these will become the

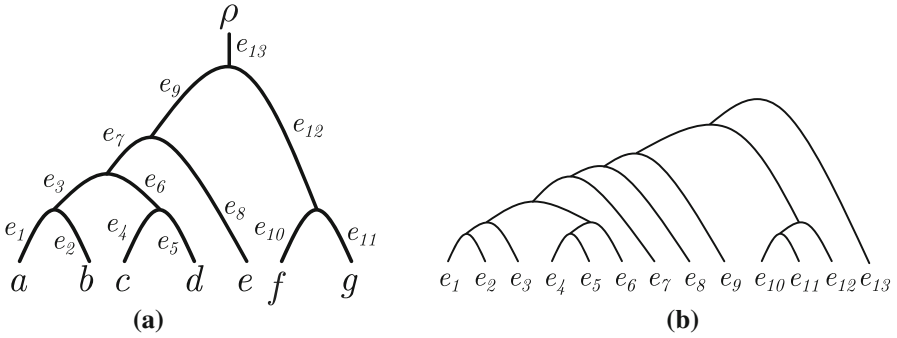


Fig. 1 **a** A phylogenetic tree and **b** a segment decomposition for it

leaves of T_D. For each iteration, we identify pairs of maximal nodes corresponding to pairs of segments which can be combined to give new segments. We make the nodes in each pair children of a new node. The process continues until one component remains and T_D is a single rooted tree.

The following Proposition shows that in any partition of $E(T_2)$ into segments we can always find a large number of pairs of disjoint segments which can be merged to give other segments.

Proposition 3 *Let T be a binary tree. Let $\mathscr{M}$ be a collection of segments which partition $E(T)$. Then there are at least $\frac{|\mathscr{M}|}{4}$ non-overlapping pairs (A, B) such that $A, B \in \mathscr{M}$ and $A \cup B$ is a segment of T.*

Proof Let $\mathscr{G}_{\mathscr{M}} = (\mathscr{V}_{\mathscr{M}}, \mathscr{E}_{\mathscr{M}})$ be the graph with vertex set

$$\mathscr{V}_{\mathscr{M}} = \bigcup_{A \in \mathscr{M}} \partial A$$

and edge set

$$\mathscr{E}_{\mathscr{M}} = \{\{u, v\} : \partial A = \{u, v\} \text{ for some } A \in \mathscr{M}\}.$$

Decompose $\mathscr{G}_{\mathscr{M}}$ into maximal paths $P_1, P_2, \ldots, P_\kappa$ with the property that for each degree three vertex v of $\mathscr{G}_{\mathscr{M}}$ and each path P_i at most one of the edges incident with v is contained in P_i. For each i, let $\mathscr{M}_i$ be the set of elements $A \in \mathscr{M}$ such that $\partial A \subseteq P_i$. The sets $\mathscr{M}_i$ partition $\mathscr{M}$.

Fix one path $P_i = v_1, v_2 \ldots, v_\ell$. We order the elements of $\mathscr{M}_i$ lexicographically with respect to the indices of their boundary vertices. In other words, if $A, B \in \mathscr{M}_i$ satisfy $\partial A = \{v_j, v_k\}$ and $\partial B = \{v_\ell, v_m\}$ (where we might have $j = k$ or $\ell = m$) then we write $A < B$ if $\max(j, k) < \max(\ell, m)$ or $\max(j, k) = \max(\ell, m)$ and $\min(j, k) < \min(\ell, m)$. With this ordering, if A_k and A_{k+1} are adjacent then $(A_k \cup A_{k+1})$ is connected and has degree at most two. Hence by pairing off A_1 and A_2, A_3 and A_4, and so on, we can construct $\lfloor \frac{|\mathscr{M}_i|}{2} \rfloor$ disjoint pairs. An example is given in Fig. 2.

The total number of pairs we obtain this way is given by $\sum_{i=1}^{\kappa} \lfloor \frac{|\mathscr{M}_i|}{2} \rfloor$. We will determine a lower bound for this sum. Let d be the number of degree three vertices in $\mathscr{G}_{\mathscr{M}}$. Since $\mathscr{G}_{\mathscr{M}}$ is connected and acyclic there are $d + 2$ paths P_i which contain a degree one vertex in $\mathscr{G}_{\mathscr{M}}$ and $d - 1$ paths which do not. If P_i contains a degree one vertex then $\mathscr{M}_i$ contains at least one component with degree two and another component with boundary equal to the degree one vertex, so $|\mathscr{M}_i| \geq 2$. If P_i contains no degree one vertices then $|\mathscr{M}_i|$ is at least one. Let x denote the number of paths P_i which contain a degree one vertex and for which $|\mathscr{M}_i|$ is odd (and hence at least three). We have

$$|\mathscr{M}| = \sum_{i=1}^{\kappa} |\mathscr{M}_i| \geq 3x + 2(d + 2 - x) + (d - 1) = x + 3d + 3$$

as well as $0 \leq x \leq d + 2$ and $d \geq 0$.

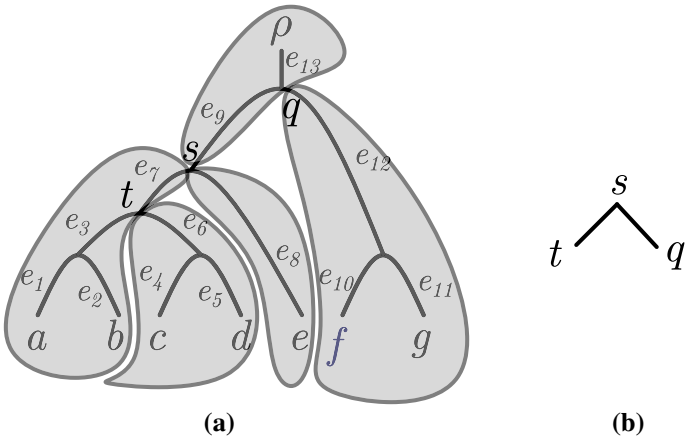


Fig. 2 **a** A phylogenetic tree with the segment decomposition $\mathscr{M} = \{\{e_1, e_2, e_3, e_7\}, \{e_4, e_5, e_6\}, \{e_8\}, \{e_9, e_{13}\}, \{e_{10}, e_{11}, e_{12}\}\}$ drawn on it, with boundary sets respectively $\{t, s\}, \{t, t\}, \{s, s\}, \{s, q\}, \{q, q\}$. **b** The corresponding $\mathscr{G}_{\mathscr{M}}$. For this decomposition, there is a single maximal path $P_1 = t, s, q := v_1, v_2, v_3$ and boundary sets become respectively $\{v_1, v_2\}, \{v_1, v_1\}, \{v_2, v_2\}, \{v_2, v_3\}, \{v_3, v_3\}$. Thus, the ordering of $\mathscr{M}$ is $\{\{e_4, e_5, e_6\}, \{e_1, e_2, e_3, e_7\}, \{e_8\}, \{e_9, e_{13}\}, \{e_{10}, e_{11}, e_{12}\}\}$

We have that $|\mathscr{M}_i|$ is even for at least $(d+2) - x$ paths ending in a degree one vertex, and for these paths $\frac{|\mathscr{M}_i|}{2} = \lfloor \frac{|\mathscr{M}_i|}{2} \rfloor$. Thus

$$\frac{|\mathscr{M}|}{2} - \sum_{i=1}^{\kappa} \left\lfloor \frac{|\mathscr{M}_i|}{2} \right\rfloor \leq \frac{x}{2} + \frac{d-1}{2}.$$

To bound the right hand side, note that the linear program

$$\begin{aligned} \max \quad & x + d \\ \text{subj. to} \quad & x - d \leq 2 \\ & x + 3d \leq |\mathscr{M}| - 3 \end{aligned}$$

has solution $d = \frac{|\mathscr{M}|-5}{4}$, $x = \frac{|\mathscr{M}|+3}{4}$ and so $x + d \leq \frac{2|\mathscr{M}|-2}{4}$. Hence

$$\frac{|\mathscr{M}|}{2} - \sum_{i=1}^{\kappa} \left\lfloor \frac{|\mathscr{M}_i|}{2} \right\rfloor \leq \frac{|\mathscr{M}|}{4} - \frac{3}{4}$$

and $\sum_{i=1}^{\kappa} \lfloor \frac{|\mathscr{M}_i|}{2} \rfloor$, the number of pairs, is bounded below by $\frac{|\mathscr{M}|}{4}$. □

We can now state the algorithm for constructing T_D. Initially T_D is a set of isolated vertices. As the algorithm progresses, vertices are combined into larger trees, so that each iteration T_D is a forest. The algorithm terminates when T_D contains a single tree.

At each iteration let $\mathscr{M}$ denote the partition of the edge set of $E(T)$ into segments corresponding to the maximal elements of the incomplete tree T_D. Rather than store

this partition explicitly, we maintain a linked list $\mathscr{B}$ of boundary nodes. For each element v in the list we maintain pointers to maximal nodes T_D corresponding to segments in $\mathscr{M}$ having v in their boundaries. In addition, we maintain pointers from each node in T_D to the boundary nodes of the corresponding segments.

1. Initialize T_D with a forest of degree-one vertices corresponding to each edge of $E(T_2)$. Hence we initialise $\mathscr{B}$ with one element for each vertex in $V(T_2)$, with the associated pointers. At this point, $\mathscr{M}$ is the partition of $E(T_2)$ putting each edge into a separate block.
2. **While** T_D is disconnected **do**
   (a) Using the construction in Proposition 3 determine a set of at least $k \geq \frac{|\mathscr{M}|}{4}$ pairs $(A_1, B_1), \ldots, (A_k, B_k)$ of disjoint elements of $\mathscr{M}$ such that $A_j \cup B_j$ has at most two boundary points.
   (b) For each pair $(A_i, B_i), i = 1, 2, \ldots, k$, create a new node of T_D corresponding to $A_i \cup B_i$ and with children corresponding to A_i and B_i.
   (c) Update the list $\mathscr{B}$ of boundary vertices and the associated pointers.

Theorem 4 *We can construct a segment decomposition tree T_D for T_2 with height $O(\log n)$ in $O(n)$ time.*

Proof We only merge nodes if the union of their corresponding segments is also a segment. Hence T_D will be a segment decomposition tree. It remains to prove the bound on height and running time.

We note that $|\mathscr{M}|$ reduces by a factor of $\frac{3}{4}$ each iteration. Hence the number of iterations is at most $\log_{\frac{4}{3}}(2n-3)$, which is also a bound on the height of the tree.

Using the list of boundary points $\mathscr{B}$ we can construct construct $\mathscr{G}_{\mathscr{M}}$ and identify pairs, in $O(|\mathscr{M}|)$ time each iteration. Thus the total running time is at most $O(n(\sum_{i=0}\left(\frac{3}{4}\right)^i)) = O(n)$ time. □

We can strengthen the height bound. We say that a tree is k-locally balanced if, for all nodes v in the tree, the height of the subtree rooted at v is at most $k \cdot (1 + \log |v|)$, where $|v|$ is the number of leaves in the subtree rooted at v. As the algorithm can be applied recursively on each node of T_D we have that the global height bound applies to each node. Hence

Corollary 5 *The segment decomposition T_D is $(1/\log\frac{4}{3})$-locally balanced.*

4 Computing the Inner Product

In this section we show that $\sum_{ij} p_{ij}q_{ij}$ can be computed in $O(n \log n)$ time, so that the main result follows from Eq. (2). Once again we make a lot of use of numerous ideas from [2], though with a modified decomposition and a 2-coloring, rather than a 3-coloring. The overall strategy of using the decomposition to updated calculations from one edge to the next efficiently is the same, and our Lemma 10 is a restatement of Lemma 7 in [2].

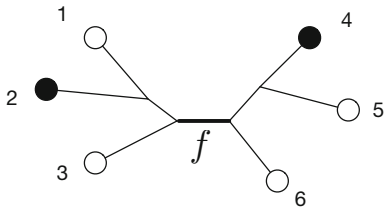


Fig. 3 Illustration of the definition of $\widetilde{\chi}(c, f)$. For the coloring indicated, the pairs of taxa which cross the edge f and have different colors are $\{1, 4\}$, $\{3, 4\}$, $\{2, 5\}$, and $\{2, 6\}$. Hence $\widetilde{\chi}(c, f) = 4$

There are many aspects of our problem which are simpler, for one thing we only need to consider 2-colorings rather than 3-colorings. Furthermore the differences in the decompositions used mean that several results need to be proved anew.

A *(taxon) coloring* is an assignment c of the colors black and white to the taxa. For each edge e of T_1 we let c_e denote a coloring assigning black to those taxa on one side of e and white to those on the other. For each edge f in $E(T_2)$ and each colouring c of the set of taxa, we let $\widetilde{\chi}(c, f)$ denote the number of pairs of taxa ij such that i and j have different colours and they label leaves on different sides of f (Fig. 3).

Lemma 6

$$\sum_{ij} p_{ij}q_{ij} = \sum_{e\in E(T_1)} \sum_{f\in E(T_2)} x_e y_f \widetilde{\chi}(c_e, f) \tag{7}$$

Proof

$$\begin{aligned}\sum_{ij} p_{ij}q_{ij} &= \sum_{ij} \left(\sum_{e:e\in A_{ij}} x_e \right) \left(\sum_{f:f\in B_{ij}} y_f \right) \\ &= \sum_{ij} \sum_{e\in A_{ij}} \sum_{f\in B_{ij}} x_e y_f \qquad (8) \\ &= \sum_{e\in E(T_1)} \sum_{f\in E(T_2)} x_e y_f \widetilde{\chi}(c_e, f). \qquad (9)\end{aligned}$$

□

For the remainder of this section we will assume that the vertices in T_2 are indexed $v_1, v_2, \ldots, v_{2n-2}$. The actual ordering does not matter; it is only used to help presentation.

Let T_D be the segment decomposition tree constructed for T_2 using the Algorithm in Sect. 3. For each node v of T_D we let $Q_v \subseteq E(T_2)$ denote corresponding segment in T_2. The overall strategy at this point is to compute values for each node in T_D which will allow us to: (i) compute, for an initial choice of $e \in E(T_1)$, the sum $\sum_{f\in E(T_2)} x_e y_f \widetilde{\chi}(c_e, f)$ in linear time, and (ii) update this computation efficiently as we iterate in a particular way through edges e of T_1.

We will store three pieces of information at every non-root node v of T_D, the exact type of information stored being dependent on the degree of the segment Q_v corresponding to v.

If Q_v is degree one then we store:

- Two integer counts w_v, b_v
- A description (e.g. coefficients) for a quadratic polynomial $\phi_v(\cdot, \cdot)$ with two variables.

If Q_v has degree two then we store:

- Two integer counts w_v, b_v
- A description (e.g. coefficients) for a quadratic polynomial $\phi_v(\cdot, \cdot, \cdot, \cdot)$ with four variables.

The exact interpretation of w_v, b_v and ϕ_v is given below as statements (C1)–(C3) in the proof of Lemma 7. The basic intuition is that v is a node in the segment decomposition, so corresponds to a segment Q_v, a subset of the edge set of T_2. The counts b_v and w_v are the numbers of leaves coloured black and white (respectively) by c_e in the segment Q_v. The polynomial ϕ_v plays a key role in the computation of $\sum_{f \in Q_v} y_f \widetilde{\chi}(c_e, f)$ which, at the top of the decomposition tree, provides $\sum_{f \in E(T_2)} x_e y_f \widetilde{\chi}(c_e, f)$.

We now show how the values b_v, w_v and ϕ_v are computed using a colouring c of the taxa. We start at the leaves of T_D and work upwards towards the root.

Suppose that v is a leaf of T_D, so that Q_v contains a single edge f of T_2. There are two cases.

1. The edge f is incident with a leaf u of T_2, so Q_v has degree one. If $c(u)$ is black then $b_v = 1$ and $w_v = 0$, while if $c(u)$ is white we have $w_v = 1$ and $b_v = 0$. In either case
$$\phi_v(b, w) = y_f(b \cdot w_v + w \cdot b_v). \tag{10}$$
2. The edge f is not incident with a leaf of T_2, so Q_v has degree two. Then $b_v = w_v = 0$ and
$$\phi_v(b_1, w_1, b_2, w_2) = (b_1 w_2 + b_2 w_1) y_f. \tag{11}$$

Now suppose that v is an internal vertex of T_D. Once again there are several cases, however in all cases we have

$$b_v = b_{v_L} + b_{v_R}$$
$$w_v = w_{v_L} + w_{v_R}.$$

3. Suppose Q_{v_L} and Q_{v_R} have degree one. Then
$$\phi_v(b, w) = \phi_{v_L}(b + b_{v_R}, w + w_{v_R}) + \phi_{v_R}(b + b_{v_L}, w + w_{v_L}). \tag{12}$$
4. Suppose Q_{v_L} has degree two and Q_{v_R} has degree one, where $\partial Q_{v_L} = \{v_i, v_j\}$ and $Q_{v_R} = \{v_j\}$.
   (a) If Q_v has degree one and $i < j$ then
   $$\phi_v(b, w) = \phi_{v_L}(b, w, b_{v_R}, w_{v_R}) + \phi_{v_R}(b + b_{v_L}, w + w_{v_L}); \tag{13}$$
   (b) If Q_v has degree one and $i > j$ then
   $$\phi_v(b, w) = \phi_{v_L}(b_{v_R}, w_{v_R}, b, w) + \phi_{v_R}(b + b_{v_L}, w + w_{v_L}); \tag{14}$$

(c) If Q_v has degree two and $i < j$ then

$$\phi_v(b_1, w_1, b_2, w_2) = \phi_{v_L}(b_1, w_1, b_2 + b_{v_R}, w_2 + w_{v_R}) + \phi_{v_R}(b_1 + b_2 + b_{v_L}, w_1 + w_2 + w_{v_L}); \quad (15)$$

(d) If Q_v has degree two and $i > j$ then

$$\phi_v(b_1, w_1, b_2, w_2) = \phi_{v_L}(b_1 + b_{v_R}, w_1 + w_{v_R}, b_2, w_2) + \phi_{v_R}(b_1 + b_2 + b_{v_L}, w_1 + w_2 + w_{v_L}). \quad (16)$$

5. The case when Q_{v_L} has degree one and Q_{v_R} has degree two is symmetric.
6. Suppose that Q_{v_L} and Q_{v_R} have degree two, that $\partial Q_{v_L} = \{v_i, v_j\}$ and $\partial Q_{v_R} = \{v_j, v_k\}$. We can assume that $i < k$ since the alternative case follows by symmetry. This leaves three possibilities:

(a) If $i < j$ and $j < k$ then

$$\phi_v(b_1, w_1, b_2, w_2) = \phi_{v_L}(b_1, w_1, b_2 + b_{v_R}, w_2 + w_{v_R}) + \phi_{v_R}(b_1 + b_{v_R}, w_1 + w_{v_R}, b_2, w_2); \quad (17)$$

(b) If $i < j$ and $j > k$ then

$$\phi_v(b_1, w_1, b_2, w_2) = \phi_{v_L}(b_1, w_1, b_2 + b_{v_R}, w_2 + w_{v_R}) + \phi_{v_R}(b_2, w_2, b_1 + b_{v_L}, w_1 + w_{v_L}); \quad (18)$$

(c) If $j < i$ and (hence) $j < k$ then

$$\phi_v(b_1, w_1, b_2, w_2) = \phi_{v_L}(b_1 + b_{v_R}, w_1 + w_{v_R}, b_1, w_1) + \phi_{v_R}(b_1 + b_{v_R}, w_1 + w_{v_R}, b_2, w_2). \quad (19)$$

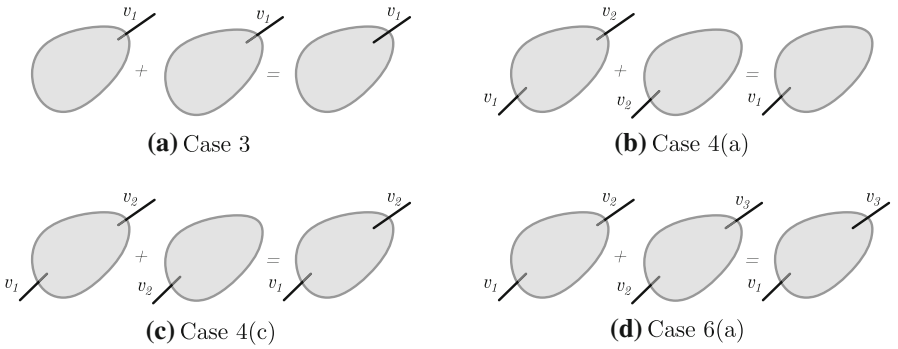
An illustration for several of these cases can be found in Fig. 4.

Lemma 7 *Suppose that b_v, w_v and ϕ_v have been computed as above for all nodes of T_D except the root. Let v_L and v_R be the children of the root of T_D. Then*

$$\sum_{f \in E(T_2)} \widetilde{\chi}(c, f) y_f = \phi_{V_L}(b_{v_R}, w_{v_R}) + \phi_{V_R}(b_{v_L}, w_{v_L}).$$

Proof For any node v of T_D we let L_v denote the set of leaves of T_2 not incident with an edge of Q_v. If Q_v has degree two and boundary $\{v_i, v_j\}$, $i < j$, then we let $L_v^{(1)}$ be the leaves in L_v which are closest to v_i and $L_v^{(2)}$ the leaves in L_v which are closest to v_j. Let $\tilde{c}$ be any colouring of the leaves of T_2, possibly distinct from c. Let B and W be the sets of leaves that $\tilde{c}$ colours black and white respectively.

We will establish the following claims for all nodes v in T_D, using induction on the height of the node.



(a) Case 3 **(b)** Case 4(a)

(c) Case 4(c) **(d)** Case 6(a)

Fig. 4 Cartoons of segment merging for several cases discussed in the main text. Grey "blobs" are used to illustrate segments and v_1, v_2, v_3 are the boundaries of these segments

(C1) b_v and w_v are the number of leaves incident with edges in Q_v which are coloured black and white by c (and hence by $\tilde{c}$).

(C2) If Q_v has degree one, $b = |B \cap L_v|$ and $w = |W \cap L_v|$ then

$$\sum_{f \in Q_v} \widetilde{\chi}(\tilde{c}, f) y_f = \phi_v(b, w).$$

(C3) If Q_v has degree two, $b_1 = |B \cap L_v^{(1)}|$, $w_1 = |W \cap L_v^{(1)}|$, $b_2 = |B \cap L_v^{(2)}|$, and $w_2 = |B \cap L_v^{(2)}|$ then

$$\sum_{f \in Q_v} \widetilde{\chi}(\tilde{c}, f) y_f = \phi_v(b_1, w_1, b_2, w_2).$$

We start by considering any leaf v of T_D. In this case, Q_v contains a single edge f. If f is an edge incident to a leaf coloured white then $b_v = 0$, $w_v = 1$ as required, and $\widetilde{\chi}(\tilde{c}, f)$ equals the number of leaves coloured black by $\tilde{c}$, so

$$\widetilde{\chi}(\tilde{c}, f) y_f = |B \cap L_v| y_f = (b w_v + w b_v) y_f = \phi_v(b, w).$$

The same holds if the leaf is coloured black.

If the edge f is internal then $b_v = w_v = 0$, and $\widetilde{\chi}(\tilde{c}, f)$ is equal to the number of paths crossing f connecting leaves with different colours, or

$$|B \cap L_v^{(1)}||W \cap L_v^{(2)}| + |W \cap L_v^{(1)}||B \cap L_v^{(2)}| = b_1 w_2 + w_1 b_2,$$

so $\widetilde{\chi}(\tilde{c}, f) y_f = \phi_v(b_1, w_1, b_2, w_2)$.

Now consider the case when v is an internal node of T_D, other than the root. Let v_L and v_R be the two children of v. Note that Q_v is the disjoint union of Q_{v_L} and Q_{v_R}, so $b_v = b_{v_L} + b_{v_R}$ and $w_v = w_{v_L} + w_{v_R}$, proving (C1).

Furthermore, we have

$$\sum_{f \in Q_v} \widetilde{\chi}(\tilde{c}, f) = \sum_{f \in Q_{v_L}} \widetilde{\chi}(\tilde{c}, f) + \sum_{f \in Q_{v_R}} \widetilde{\chi}(\tilde{c}, f).$$

If Q_{v_L} has degree one then, by the induction hypothesis,

$$\sum_{f \in Q_{v_L}} \widetilde{\chi}(\tilde{c}, f) = \phi_{v_L}(b', w')$$

where b' and w' are the numbers of leaves coloured black and white that are not incident with edges in Q_{v_L}. Similarly, if Q_{v_L} has degree two then, by the induction hypothesis,

$$\sum_{f \in Q_{v_L}} \widetilde{\chi}(\tilde{c}, f) = \phi_{v_L}(b'_1, w'_1, b'_2, w'_2)$$

where b'_1 and w'_1 are the numbers of leaves coloured black and white that are not incident with edges in Q_{v_L} and are closer to the boundary vertex of Q_{v_L} with the smallest index, while b'_2 and w'_2 are the numbers of leaves coloured black and white that are not incident with edges in Q_{v_L} and are closer to the boundary vertex of Q_{v_L} with the largest index. The symmetric result holds for Q_{v_R}.
The different cases in Eqs. (12)–(19) now correspond to the different counts for b', w' or for b'_1, w'_1, b'_2, w'_2 depending on whether Q_{v_L} and Q_{v_R} have degree one or two, and whether the boundary vertices in common had the highest or lower index for each segment.

Now suppose that v_L and v_R are the children of the root of T_B. Then $\partial(Q_{v_L} \cup Q_{v_R}) = \emptyset$ so Q_{v_L} and Q_{v_R} must both have degree one. We have that $E(T_2)$ is the disjoint union of Q_{v_L} and Q_{v_R}. Any leaf not incident to a leaf in Q_{v_L} is incident to a leaf in Q_{v_R} and vice versa. Hence as required. □

Evaluating Eqs. (12)–(19) takes constant time and space per each node of T_D, since we manipulate and store a constant number of polynomials with at most four variables and total degree at most two. Thus, evaluating Eqs. (12)–(19) takes $O(n)$ time and space for each colouring, and since we want to sum this quantity over all colourings c_e from edges $e \in E(T_1)$ a naive implementation would still take $O(n^2)$ time. The key to improving this bound is in the use of efficient updates. We make use of the following upper bound in [2]. Recall from Corollary 5 that the segment decomposition tree T_D we construct is T_D is $(1/\log \frac{4}{3})$-locally balanced.

Lemma 8 *The union of k root-to-leaf paths in a c-locally balanced rooted binary tree with n leaves contains at most $k(3 + 4c) + 2ck \log(n/k)$ nodes.*

Proof This is Lemma 2 in [2].

Lemma 9 *Suppose that we have computed b_v, w_v and the functions ϕ_v for all $v \in T_D$, using a leaf colouring c. Let $\tilde{c}$ be a colouring which differs from c at k leaves. Then we can update the values b_v, w_v and the functions ϕ_v in $O(k + k \log(n/k))$ time.*

procedure SUM(u)
  Let e be the edge connecting u to its parent (in T_1).
  $x \leftarrow \sum_{f \in E(T_2)} x_e y_f \widetilde{\chi}(c, f)$, computed using T_D.
  if u is a leaf **then**
    Color u black and update T_D
    return x
  else
    Color the leaves in the subtree of T_1 rooted at Small(u) in black and update T_D
    $y \leftarrow$ SUM(Large(u))
    Color the leaves in the subtree of T_1 rooted at Small(u) in white and update T_D
    $z \leftarrow$ SUM(Small(u))
    return $x + y + z$
  end if
end procedure

Fig. 5 Recursive algorithm SUM

Proof Let F' be the set of edges of T_2 which are incident to a leaf for which c and $\tilde{c}$ have a different colour, so $|F'| = k$. The only nodes v in T_D which need to updated are those with $f \in Q_v$ for some $f \in F'$. This is a union of the paths from k leaves of T_D to the root of T_D, and so by Lemma 8, it has size $O(k + k \cdot \log(\frac{n}{k}))$. □

The final step is to show that we can navigate the edges in $E(T_1)$ so that the total number of changes in the colourings is bounded appropriately. Suppose that T_1 is rooted at the leaf ρ (the same as T_2). For each internal node u in T_1 we let Small(u) denote the child of u with the smallest number of leaf descendants and let Large(u) denote the child with the largest number of leaf descendants, breaking ties arbitrarily.

The recursive function SUM(u) in Fig. 5 returns the sum of

$$\sum_{f \in E(T_2)} x_e y_f \widetilde{\chi}(c_e, f)$$

over all edges $e \in E(T_1)$. Initially we let e be the edge incident with the root ρ. Let c be the colouring where ρ is black and all other leaves white. We initialise T_D and fill out the values b_v, w_v and ϕ_v for all nodes v of T_D using the colouring c. We then call SUM(u) where u is the unique internal node adjacent to ρ.

We see that the algorithm makes a pre-order traversal of T_1, evaluating the sum

$$\sum_{f \in E(T_2)} x_e y_f \widetilde{\chi}(c_e, f)$$

for each edge e and accumulating the total. Thus by Lemma 6, the algorithm returns $\sum_{ij} p_{ij} q_{ij}$.

The running time is dominated by the time required to update T_D. For each leaf, the update is made after only one leaf changes colour, so this takes $O(n \log n)$ summed over all leaves. For every other node u in the tree, the number of nodes of T_D to update

is $O(k + k\log(n/k))$ where k is the number of leaves in the subtree rooted at Small(u) Lemma 8.

Lemma 10 *Let T be a rooted binary tree with n leaves and for each internal node u of T let k_u denote the number of leaves in the smallest subtree rooted at a child of u. Then*

$$\sum_{u \text{ internal}} k_u \log(n/k_u) \leq n \log n.$$

Proof This is a restatement of Lemma 7 in [2]. □

Theorem 11 *Algorithm* SUM *computes $\sum_{ij} p_{ij}q_{ij}$ in $O(n \log n)$ time. Hence the path length distance between T_1 and T_2 can be computed in $O(n \log n)$ time.*

Acknowledgements This research was made possible due to travel funds made available from a Marsden Grant to DB. Both authors thank David Swofford for help finding an error in an earlier version of Proposition 1.

References

1. Brent, R.P.: The parallel evaluation of general arithmetic expressions. J. ACM (JACM) **21**(2), 201–206 (1974)
2. Brodal, G.S., Fagerberg, R., Pedersen, C.N.: Computing the quartet distance between evolutionary trees in time $O(n \log n)$. Algorithmica **38**(2), 377–395 (2004)
3. Bryant, D.: A classification of consensus methods for phylogenetics. DIMACS Ser. Discrete Math. Theor. Comput. Sci. **61**, 163–184 (2003)
4. Bryant, D., Waddell, P.: Rapid evaluation of least squares and minimum evolution criteria on phylogenetic trees. Mol. Biol. Evol. **15**(10), 1346–1359 (1997)
5. Cohen, R.F., Tamassia, R.: Dynamic expression trees. Algorithmica **13**(3), 245–265 (1995)
6. Farris, J.S.: A successive approximations approach to character weighting. Syst. Biol. **18**(4), 374–385 (1969)
7. Hartigan, J.A.: Representation of similarity matrices by trees. J. Am. Stat. Assoc. **62**(320), 1140–1158 (1967)
8. Hillis, D.M., Heath, T.A., John, K.S.: Analysis and visualization of tree space. Syst. Biol. **54**(3), 471–482 (2005)
9. Holmes, S.: Statistical approach to tests involving phylogenies. In: Gascuel, O. (ed.) Mathematics of Phylogeny and Evolution, chap. 4, pp. 91–117. New York: Oxford University Press (2005)
10. Lapointe, F.J., Cucumel, G.: The average consensus procedure: combination of weighted trees containing identical or overlapping sets of taxa. Syst. Biol. **46**(2), 306–312 (1997)
11. Penny, D., Watson, E.E., Steel, M.A.: Trees from languages and genes are very similar. Syst. Biol. **42**(3), 382–384 (1993)
12. Robinson, D., Foulds, L.: Comparison of phylogenetic trees. Math. Biosci. **53**, 131–147 (1981)
13. Susko, E.: Improved least squares topology testing and estimation. Syst. Biol. **60**(5), 668–675 (2011)
14. Swofford, D.L.: When are phylogeny estimates from molecular and morphological data incongruent? In: Miyamoto, M.M., Cracraft, J. (eds.) Phylogenetic Analysis of DNA Sequences, pp. 295–333. Oxford University Press, Oxford (1991)
15. Williams, W.T., Clifford, H.T.: On the comparison of two classifications of the same set of elements. Taxon **20**(4), 519–522 (1971)