Calculating the Unrooted Subtree Prune-and-Regraft Distance

Chris Whidden[ID] and Frederick A. Matsen IV[ID]

Abstract—The subtree prune-and-regraft (SPR) distance metric is a fundamental way of comparing evolutionary trees. It has wide-ranging applications, such as to study lateral genetic transfer, viral recombination, and Markov chain Monte Carlo phylogenetic inference. Although the rooted version of SPR distance can be computed relatively efficiently between rooted trees using fixed-parameter-tractable maximum agreement forest (MAF) algorithms, no MAF formulation is known for the unrooted case. Correspondingly, previous algorithms are unable to compute unrooted SPR distances larger than 7. In this paper, we substantially advance understanding of and computational algorithms for the unrooted SPR distance. First, we identify four properties of optimal SPR paths, each of which suggests that no MAF formulation exists in the unrooted case. Then, we introduce the replug distance, a new lower bound on the unrooted SPR distance that is amenable to MAF methods, and give an efficient fixed-parameter algorithm for calculating it. Finally, we develop a "progressive A*" search algorithm using multiple heuristics, including the TBR and replug distances, to exactly compute the unrooted SPR distance. Our algorithm is nearly two orders of magnitude faster than previous methods on small trees, and allows computation of unrooted SPR distances as large as 14 on trees with 50 leaves.

Index Terms—Fixed-parameter tractability, phylogenetics, subtree prune-and-regraft distance, lateral gene transfer, agreement forest

✦

1 INTRODUCTION

MOLECULAR phylogenetic methods reconstruct evolutionary trees (a.k.a phylogenies) from DNA or RNA data and are of fundamental importance to modern biology [1]. Phylogenetic inference has numerous applications including investigating organismal relationships (the "tree of life" [2]), reconstructing virus evolution away from innate and adaptive immune defenses [3], analyzing the immune system response to HIV [4], designing genetically-informed conservation measures [5], and investigating the human microbiome [6]. Although the molecular evolution assumptions may differ for these different settings, the core algorithmic challenges remain the same: reconstruct a tree graph representing the evolutionary history of a collection of evolving units, which are abstracted as a collection of *taxa*, where each *taxon* is associated with a DNA, RNA, or amino acid sequence.

Phylogenetic study often requires an efficient means of comparing phylogenies in a meaningful way. For example, different inference methods may construct different phylogenies and it is necessary to determine to what extent they differ and, perhaps more importantly, which specific features differ between the trees. In addition, the evolutionary history of individual genes does not necessarily follow the overall history of a species due to *reticulate* evolutionary processes: lateral genetic transfer, recombination, hybridization, and

incomplete lineage sorting [7]. Such processes impact phylogenies by moving a subtree from one location to another, as described below. Thus comparing inferred histories of genes to each other, a reference tree, or a proposed species tree may be used to identify reticulate events [8], [9]. Moreover, distance measures between phylogenies provide optimization criteria that can be used to infer summary measures such as supertrees [9], [10], [11], [12].

Numerous distance measures have been proposed for comparing phylogenies. The Robinson-Foulds distance [13] is perhaps the most well known and can be calculated in linear time [14]. However, the Robinson-Foulds distance has no meaningful biological interpretation or relationship to reticulate evolution. Typically, distance metrics are either easy to compute but share this lack of biological relation, such as the quartet distance [15] and geodesic distance [16], or are difficult to compute, such as the hybridization number [17] and maximum parsimony distance [18], [19], [20].

The subtree prune-and-regraft (SPR) distance is widely used due to its biological interpretability despite being difficult to compute [17], [21]. SPR distance is the minimum number of subtree moves required to transform one tree into the other (Fig. 1d). It provides a lower bound on the number of reticulation events required to reconcile two phylogenies. As such, it has been used to model reticulate evolution [22], [23]. In addition, the SPR distance is a natural measure of distance when analyzing phylogenetic inference methods which typically apply SPR operations to find maximum likelihood trees [24], [25] or estimate Bayesian posterior distributions with SPR-based Metropolis-Hastings random walks [26], [27]. Similar trees can be easily identified using the SPR distance, as random pairs of n-leaf trees differ by an expected $n - \Theta(n^{2/3})$ SPR moves [28]. This difference approaches the maximum SPR distance of

- *The authors are with the Program in Computational Biology, Fred Hutchinson Cancer Research Center, Seattle, WA 98109.*
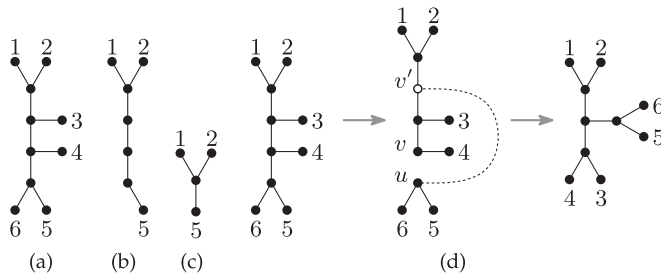*E-mail: {cwhidden, matsen}@fredhutch.org.*

Fig. 1. (a) An unrooted X-tree T. (b) $T(V)$, where $V = \{1, 2, 5\}$. (c) $T|V$. (d) An SPR operation transforms T into a new tree by *pruning a subtree* and *regrafting* it in an other location.

$n - 3 - \lfloor (\sqrt{n-2} - 1)/2 \rfloor$ asymptotically [29]. The topology-based SPR distance is especially appropriate in this context as topology changes have been identified as the main limiting factor of such methods [30], [31], [32]. Moreover, the SPR distance has close connections to network models of evolution [17], [23], [33].

Although it has these advantages, the SPR distance between both rooted and unrooted trees is NP-hard to compute [34], [35], limiting its utility. We recall that *rooted* trees represent the usual view of evolution, such that the taxa under consideration evolve from a common ancestor in a known direction. *Unrooted* trees drop this implied directionality, and are typically drawn as a graph theoretic tree such that every non-leaf node has degree three (Fig. 1a). As described below, most phylogenetic algorithms reconstruct unrooted trees.

Despite the NP-hardness of computing the SPR distance between rooted phylogenies, recent algorithms can rapidly compare rooted trees with hundreds of leaves and SPR distances of 50 or more in fractions of a second [9], [36]. This has enabled use of the SPR distance for inferring phylogenetic supertrees and lateral genetic transfer [9], comparing influenza phylogenies to assess reassortment [37], and investigating mixing of Bayesian phylogenetic posteriors [32], [38]. However, most phylogenetic inference packages today use reversible mutation models to infer unrooted trees, motivating SPR calculation for unrooted trees.

SPR distances can be computed efficiently in practice for rooted trees for two key reasons. First, they can be computed using a maximum agreement forest (MAF) of the trees [39], [40]. An MAF is a forest (i.e. collection of trees) that can be obtained from both trees by removing a minimum set of edges. Each removed edge corresponds to one SPR operation, and a set of SPR operations transforming one tree into the other can be easily recovered given an MAF. Due to this MAF framework, the development of efficient fixed-parameter and approximation algorithms for SPR distances between rooted trees has become an area of active research [21], [41], [42], [43], [44], [45] (see [43] for a more complete history), including recent extensions to non-binary trees [46], [47], as well as generalized MAFs of multiple trees [48]. The utility of MAFs motivates defining a variant definition in the unrooted case with analogous properties for unrooted SPR, however, no alternative MAF formulation has yet been developed. A straightforward extension of MAFs to unrooted trees is equivalent to a different metric, the tree-bisection-reconnection (TBR) distance [40]. Although TBR rearrangements are used in some phylogenetic inference methods, SPR rearrangements are

much more common [30] and the TBR distance does not have the other benefits of the SPR distance.

The second class of optimizations used by efficient rooted SPR algorithms are preprocessing reduction rules including the subtree reduction rule [40], chain reduction rule [40], and cluster reduction rule [49]. The subtree reduction rule also applies to unrooted trees [40] and we recently showed that the chain reduction rule is applicable to the unrooted case [50], thereby obtaining a linear-size problem kernel for unrooted SPR. However, minimum-length uSPR paths have been shown to break common clusters [35], so the cluster reduction, which partitions the trees into smaller independently solvable subproblems, is not applicable.

For all of these reasons, the best previous algorithm for computing the SPR distance between unrooted trees, due to Hickey et al. [35], cannot compute distances larger than 7 or reliably compare trees with more than 30 leaves. In this paper, we substantially advance understanding of and computational algorithms for the unrooted SPR distance. Building on previous work by Hickey et al. [35], Bonet and St. John [51], and our recently introduced "socket agreement forest (SAF)" framework [50], we make the following contributions:

(1) we identify new properties of minimum-length SPR paths showing that an MAF-like formulation is unlikely to exist,

(2) we develop a practical algorithm for enumerating maximal unrooted AFs,

(3) we define a new *replug* distance, which does admit a MAF-like formulation and gives a lower bound on the uSPR distance; we develop an exact fixed-parameter bounded search tree algorithm for its calculation, and

(4) we propose and implement a new incremental heuristic search algorithm called *progressive A** that leverages multiple increasingly expensive to compute but more accurate lower bound estimators to compute the uSPR distance in practice for trees with up to 50 leaves and distances as large as 14.

Proofs of our lemmas and theorems are in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2018.2802911.

2 PRELIMINARIES

Nodes (i.e., vertices) of a tree graph with one neighbor are called *leaves* and nodes with three neighbors are called *internal nodes*. An (unrooted binary phylogenetic) X-tree is a tree T whose nodes each have one or three neighbors, and such that the leaves of T are bijectively labeled with the members of a label set X. $T(V)$ is the unique subtree of T with the fewest nodes that connects all nodes in V. *Suppressing* a node v of degree 1 or 2 deletes v and its incident edges; if v is of degree 2 with neighbors u and w, u and w are reconnected using a new edge (u, w). The V-tree *induced* by T is the unique smallest tree $T|V$ that can be obtained from $T(V)$ by suppressing unlabeled nodes with fewer than three neighbors. See Fig. 1.

A *rooted X-tree* is defined similarly to an unrooted X-tree, with the exception that one of the internal nodes is called the *root* and is adjacent to a leaf labeled ρ. Note that this differs from the standard definition of a rooted tree in which the root is a degree two internal node. This ρ node

represents the position of the original root in a forest of the trees, as we describe below, and can simply be attached to such a degree two internal node. The parent of a node in an rooted tree is its closest neighbor to the root; the other two neighbors of an internal node are referred to as children.

An *unrooted X-forest F* is a collection of (not necessarily binary) trees $T_1, T_2, \ldots, T_k$ with respective label sets $X_1, X_2, \ldots, X_k$ such that X_i and X_j are disjoint, for all $1 \le i \ne j \le k$, and $X = X_1 \cup X_2 \cup \ldots \cup X_k$. We say that F *yields* the forest with components $T_1|X_1, T_2|X_2, \ldots, T_k|X_k$, in other words, this forest is the smallest forest that can be obtained from F by suppressing unlabeled nodes with less than three neighbors. In the rooted case, each component T_i is rooted at the node that was closest to ρ. Note that only the root of T_1 is adjacent to leaf ρ. If $T_1, T_2, \ldots, T_k$ are all binary then the remaining roots have degree 2. For an edge set E, $F - E$ denotes the forest obtained by deleting the edges in E from F and $F \div E$ the yielded forest. We say that $F \div E$ is *a forest of F*. For nodes a and b of F, we will say that a can be reached from b, or $a \sim_F b$, when there is a path of edges between a and b in F. The opposite will be denoted $a \nsim_F b$.

A *subtree-prune-regraft* (uSPR) operation on an unrooted *X-tree T* cuts an edge $e = (u, v)$. This divides T into subtrees T_u and T_v, containing u and v, respectively. Then it introduces a new node v' into T_v by subdividing an edge of T_v, and adds an edge (u, v'). Finally, v is suppressed (Fig. 1d). We distinguish between SPR operations on rooted trees and unrooted trees (uSPR operations). SPR operations on rooted trees have the additional requirement that $u \ne \rho$ and that v is u's parent rather than an arbitrary neighbor of u. Note that if the node v' introduced in the rooted tree is adjacent to ρ then v' becomes the root.

A *tree-bisection-reconnection* operation on an unrooted tree is defined similarly to a uSPR operation, except that a new node u' is also introduced into T_u bisecting any edge, the added edge is (u', v') rather than (u, v'), and both u and v are suppressed. Note that uSPR operations are a subset of TBR operations, as a TBR operation may reintroduce the same endpoint on one side of the edge.

We often consider a sequence of operations applied to a tree T_1 that result in a tree T_2. These operations can be thought of as "moving" between trees and are also referred to as *moves* (e.g., an SPR move). A sequence of moves $M = m_1, m_2, \ldots, m_d$ applied to T_1 result in the sequence of trees $T_1 = t_0, t_1, t_2, \ldots, t_d = T_2$. We call such sequences of trees a *path* (e.g., an SPR path).

When considering how the tree changes throughout such sequences, it is often helpful to consider how nodes and edges of the tree change. Formally, we construct a mapping $\varphi_{i,j}$ that maps nodes and edges of t_i to t_j. Each mapping $\varphi_{i,i+1}$ between adjacent trees is constructed according to the corresponding move m_{i+1}: nodes and edges of t_i that are not modified by m_{i+1} are mapped to the corresponding nodes and edges of t_{i+1}. The deleted edge (u, v) of t_i is mapped to the newly introduced edge of t_{i+1} (e.g., (u, v') for an SPR move). Deleted nodes are mapped to $\emptyset$. Forward mappings $\varphi_{i,j}$, $i < j$, are constructed transitively. Reverse mappings $\varphi_{j,i}$, $i < j$, are constructed analogously by considering the application of moves that construct the reverse sequence $t_d, t_{d-1}, \ldots, t_0$.

We will use these mappings implicitly to talk about how a tree changes throughout a sequence of moves.
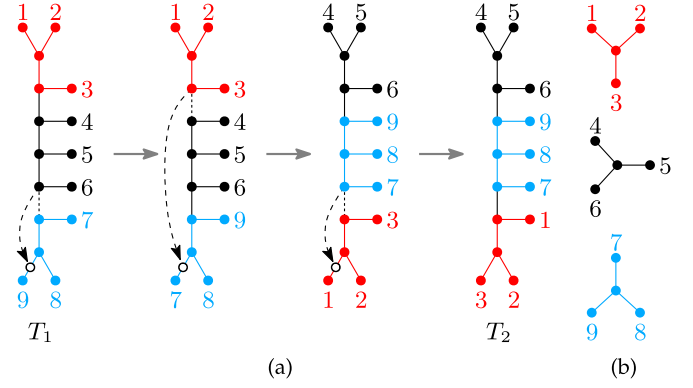

Fig. 2. (a) Three unrooted SPR operations transform a tree T_1 into another tree T_2. The dashed line with an arrow is the edge that will be added in the next step; simultaneously the dotted edge touching the tail end of the arrow will be removed. (b) A MAF of T_1 and T_2.

With these mappings we can consider SPR and TBR tree moves as changing the endpoints of edges rather than deleting one edge and introducing another. We say that an edge is *broken* if one of its endpoints is moved by a rearrangement operation. The relation between SPR and TBR moves can now be summarized by considering that a uSPR operation changes one endpoint of an edge, a rooted tree SPR operation changes the root-most endpoint of an edge, and a TBR operation changes both endpoints.

SPR operations give rise to a distance measure $d_{\mathrm{SPR}}(\cdot, \cdot)$ between X-trees, defined as the minimum number of SPR operations required to transform one tree into the other. The trees in Fig. 2a, for example, have SPR distance $d_{\mathrm{SPR}}(T_1, T_2) = 3$. The TBR distance $d_{\mathrm{TBR}}(\cdot, \cdot)$ is defined analogously with respect to TBR operations. A minimum-length path of SPR or TBR moves between two trees is an optimal SPR path or optimal TBR path, respectively.

The second and third SPR operations applied in Fig. 2a move both ends of a single edge and can be replaced by a single TBR operation. Thus, in this case $d_{\mathrm{TBR}}(T_1, T_2) = 2$. Note that the fact that uSPR operations are a subset of TBR operations implies that the TBR distance is a lower bound on the uSPR distance [40]:

Lemma 2.1. *For two unrooted trees T_1 and T_2, $d_{\mathrm{TBR}}(T_1, T_2) \le d_{\mathrm{SPR}}(T_1, T_2)$.*

Given trees T_1 and T_2 and forests F_1 of T_1 and F_2 of T_2, a forest F is an *agreement forest* (AF) of F_1 and F_2 if it is a forest of both forests. F is a *maximum agreement forest* if it has the smallest possible number of components. We denote this number of components by $m(F_1, F_2)$. For two unrooted trees T_1 and T_2, Allen and Steel [40] showed that $d_{\mathrm{TBR}}(T_1, T_2) = m(T_1, T_2) - 1$. Fig. 2b shows an MAF of the trees in Fig. 2a.

For two *rooted* trees T_1 and T_2, Bordewich and Semple [34] showed $d_{\mathrm{SPR}}(T_1, T_2) = m(T_1, T_2) - 1$, by introducing the root node augmentation ρ described above. Moreover, nodes are only suppressed in rooted trees if they have fewer than two children, such that unlabeled *component roots* (the nodes that were connected to the ρ component by an edge before cutting) are not suppressed.

In contrast to an MAF, a *maximal AF (mAF) F^** of two trees T_1 and T_2 is an AF of T_1 and T_2 that is not a forest of any other AF of T_1 and T_2. In other words, no edges can be

added to F^* to obtain an AF with fewer components. Every MAF is a mAF, but not necessarily vice versa [52].

3 FOUR STRIKES AGAINST AN AGREEMENT FOREST FOR THE uSPR DISTANCE

There are relatively efficient fixed-parameter algorithms for each distance metric with a maximum (acyclic) agreement forest formulation—the rooted SPR distance [39], unrooted TBR distance [40], and rooted hybridization number [17]. We call these MAF-like problems. For MAF-like problems, the distance can be found from easily determined properties of the maximum agreement forest: typically, its number of components. It is reasonable to ask why there has been no formulation of the uSPR distance as an MAF-like problem. In this section, we identify four properties of the uSPR distance that are contrary to natural MAF assumptions. Specifically, these properties show the uSPR distance isn't easily calculated from properties of a traditionally defined agreement forest.

Let S be an optimal sequence of SPR operations transforming one unrooted tree T_1 to another T_2. Consider the set of broken edges E of T_1 and T_2 whose endpoints are modified by applying S. Then we can naturally define the AF underlying S as $T_1 \div E = T_2 \div E$, that is the maximal forest of edges which are not modified by applying S. Note that this AF is not necessarily maximum. We can similarly define the AF underlying an optimal sequence of TBR moves.

Property 3.1. *One or both ends of an edge may move.*

An AF with $k + 1$ components represents a set of k TBR moves, each joining two components with respect to T_2. However, SPR moves only move one end of an edge and S may require both ends of a particular edge to move (e.g., the edge initially between 3 and 4 in Fig. 2). As such, an optimal SPR path may include one or two rearrangements corresponding to the same broken edge. In fact, as we show in the next property, optimal SPR paths may require three or more moves corresponding to the same broken edge.

Property 3.2. *The same endpoint of an edge may move twice.*

A useful feature of MAF-like problems is that each optimal move joins two components in the underlying AF. The minimum distance is thus one less than the number of MAF components, and it is easy to recover an optimal sequence of moves from the MAF.

However, optimal sequences of uSPR moves with respect to a given underlying agreement forest are not guaranteed to join AF components at each step. Consider, for example, the pair of trees in Fig. S3, available in the online supplemental material. These trees have only one MAF, but each of their optimal SPR paths begins by applying a move that does not join an underlying AF component. We verified this by computing the SPR distance between the second tree and each neighbor of the first tree. Moreover, we exhaustively tested each optimal SPR path underlain by the MAF and found that each such path moves the same endpoint of some edge twice. In other words, a broken edge may be moved three or more times in an optimal SPR path. Thus, even given the AF underlying an (unknown) optimal sequence of SPRs, it is not clear how to determine the sequence of SPRs or even their

number. We call this the *AF-move-recovery problem*, and suspect that the uSPR version of the problem may be NP-hard in its own right. This is in stark contrast to the trivially solved AF-move-recovery problem for MAF-like problems.

Property 3.3. *Common clusters are not always maintained.*

A *common cluster* is a set of taxa $L \subset X$ from two X-trees, T_1 and T_2, such that $T_i|L$ and $T_i|(X \setminus L)$ are disjoint and connected by a single edge for $i = 1, 2$. In MAF-like problems, it is never necessary to move taxa from one side of a common cluster edge to the other. In fact, these problems can be decomposed into pairs of common clusters which are solved independently [49], [53]. Such cluster decompositions greatly decrease the computational effort required to solve MAF-like problems, as algorithms to do so scale exponentially with the distance computed within a cluster rather than the total distance [9].

However, as previously shown by Hickey et al. [35], there exist pairs of unrooted trees such that every optimal SPR path violates a common cluster. This lack of independence between clusters is another sign that uSPR differs from MAF-like problems.

Property 3.4. *Common paths may be broken.*

In fact, the situation is even worse than identified by Hickey et al. [35]. Consider an optimal sequence M of SPR operations transforming one tree, T_1, into another, T_2, and the underlying agreement forest $F = T_1 \div E_1 = T_2 \div E_2$ for some E_1 and E_2. We say that two paths of edges $p_1 \in T_1$ and $p_2 \in T_2$ are *common paths* with respect to M if they connect the same AF components C_1 and C_2, that is $T_1 \div (E_1 \setminus p_1) = T_2 \div (E_2 \setminus p_2)$. Not only is it sometimes necessary to move taxa from one side of a cluster edge to another, it may be necessary to break a common path, as shown in Fig. S4, available in the online supplemental material. This is especially surprising, as a later SPR operation must reform the common path. However, breaking such a common path may free up sets of moves that would be otherwise impossible.

This observation implies that every AF underlying an optimal set of SPR operations may be a strict subforest of another AF, that is, every such underlying AF may not be maximal. This is in stark contrast to MAF-like problems where every underlying AF is either maximum or (for rooted hybridization number) maximal.

In summary, we believe that these properties make it unlikely that any uSPR MAF formulation is possible. The first two properties show that even if the correct edges are identified that need to be modified, it does not appear straightforward to find the optimal sequence of modifications. The second two properties show that we cannot even assume that edges found in each tree will be preserved in an optimal uSPR path. Thus, we require a different strategy.

4 SOCKET AGREEMENT FORESTS

Recently, we proposed a new type of agreement forest, socket agreement forests [50] which we summarize here. Note that SAFs are required only for Observation 5.2 and in Section 6.3, so this section can be skipped on a first read. SPR operations on general trees remove and introduce internal nodes, making it difficult to describe equivalence of sets of moves. SAFs solve this difficulty by including a finite set

of predetermined "sockets" which are the only nodes that can be involved in SPR operations and are never deleted or introduced. However, due to this fixed nature, SAFs are unsuitable for determining the SPR distance metric directly.

A *socket forest* is a collection of unrooted trees with special nodes, called *sockets*. Socket forests have special edges called *connections* that must be between two sockets. A collection of them is a *connection set*. Connections are not allowed to connect a socket to itself, although multiple connections to the same socket are allowed.

We will also use the following terminology on socket forests. A *move* is the replacement of one connection in a connection set for another. A *replug move* is a move that only changes one socket of a given connection. An *SPR move* for a given connection set is a replug move that does not create cycles. We say that a move *breaks* the replaced connection.

The *underlying* forest of a socket forest F is the forest F^* obtained from F by deleting all connections and suppressing all unconnected sockets. A socket forest F *permits an unrooted tree T* if it is possible to add connections between the sockets of F, resolve any multifurcations in some way, and suppress unconnected sockets to obtain T. We call a connected socket forest with such a connection set a *configuration* of F (e.g., a T configuration of F). Moreover, a socket forest F *permits an SPR path* if each intermediate tree along the path is permitted by F. Given two trees T_1 and T_2, a *socket agreement forest* is a socket forest that permits both T_1 and T_2. Note that the underlying forest of an SAF is an AF of T_1 and T_2.

Let $M = m_1, m_2, \ldots, m_k$ be a sequence of moves transforming tree T_1 into tree T_2 via an SAF F. We can consider the sequence of trees $T_1, t_1, t_2, \ldots, t_k = T_2$ induced by these moves, that is the sequence of tree configurations obtained by applying M to a T_1 configuration of F that results in a T_2 configuration of F. We thus discuss sockets and connections in the trees, as shorthand for the sockets and connections in the corresponding socket forest configurations.

Socket forests allow us to be precise concerning how connections change during a sequence of moves, because nodes are not deleted or introduced. Each socket can be separately identified (e.g. with a numbering), so any connection can be described irrespective of the other connections in a socket forest. As with moves on general trees, we consider the deletion and insertion of a connection as simply changing the endpoint of the connection. As such, the "new" connection maintains the same identifier. Thus, we can identify changes in a connection by the changes in the sockets it connects, again irrespective of the other connections in a socket forest. This implies a well defined notion of equivalence of moves: two moves are equivalent if they both attach a given endpoint of the same connection to the same socket. For example, we can uniquely describe a move as changing the second endpoint of connection c to socket v', regardless of the current state of the socket forest. We say that a move is *valid* for a socket forest configuration if it can be applied to that configuration. Similarly, a rearranged and/or modified sequence is valid if it is a sequence of valid moves.

Given an AF F' of two trees T_1 and T_2, we say that an SPR path between T_1 and T_2 is *optimal with respect to F'* if there exists no shorter SPR path between T_1 and T_2 where each intermediate tree along the path is permitted by F'. Socket agreement forests are a partial analogue of maximum

agreement forests: if we can construct a socket agreement forest for T_1 and T_2, we can be assured of a valid SPR path between T_1 and T_2 that is optimal with respect to the underlying agreement forest. However, it is not trivial to calculate the length of the SPR path between trees for a given socket agreement forest, and thus they are only a partial analogue of rooted maximum agreement forests. We prove the following lemma in [50].

Lemma 4.1. *Let F be a socket agreement forest of two trees T_1 and T_2. Then there exists an SPR path between T_1 and T_2 that is permitted by F and optimal with respect to the AF F^* underlying F.*

We can define an optimal replug path with respect to an AF F' in an analogous manner. It is then straightforward to apply the proof of Lemma 4.1 to replug moves to obtain the following corollary:

Corollary 4.2. *Let F be a socket agreement forest of two trees T_1 and T_2. Then there exists a replug path between T_1 and T_2 that is permitted by F and optimal with respect to the AF F^* underlying F.*

5 THE REPLUG DISTANCE AND MAXIMUM ENDPOINT AGREEMENT FORESTS

In this section, we introduce the *replug distance*, which lies between the TBR distance and SPR distance, and develop an agreement forest variant, called an *endpoint agreement forest*, for its calculation. This notion of agreement forest records the position of broken edges for unrooted trees, and thus does not have the pathologies of unrooted SPR agreement forests described in the first section.

The replug distance on trees is defined in terms of the *replug operation*, inspired by replug moves on socket forests. A replug operation is an SPR operation that does not necessarily result in a tree. That is, a replug operation again cuts an edge $e = (u, v)$ of a tree T, dividing T into T_u and T_v, but the new node v' attached to u may be chosen from either T_u or T_v. If $v' \in T_v$ then the replug operation is identical to an SPR operation. If, however, $v' \in T_u$, then adding the edge u, v' results in a disconnected graph with two components, one of which is cyclic. We further extend this operation to apply to arbitrary partially-labeled graphs, such as the result of a replug operation on a tree (Fig. 3a).

Given two trees, T_1 and T_2, the replug distance, $d_R(T_1, T_2)$, is the minimum number of replug moves required to transform T_1 into T_2. By dropping the requirement that each intermediate move result in a tree, we achieve an MAF-like problem that can be computed using a structure we call an *endpoint agreement forest* (EAF). The replug distance can be used as an approximation and lower bound of the SPR distance which, as we discuss in Section 7, can be used in an incremental heuristic search for the exact SPR distance. We show that it is fixed-parameter tractable in Theorem 6.9, but conjecture:

Conjecture 5.1. *The replug distance between two trees T_1 and T_2 is NP-hard to compute.*

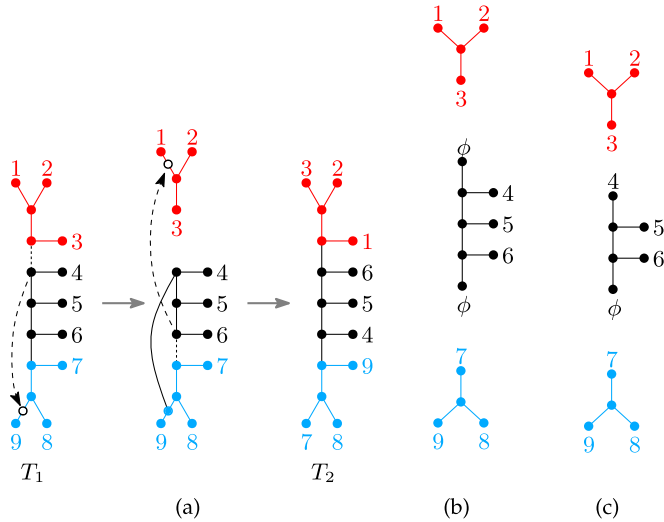To develop an agreement forest that represents replug operations, we must consider the properties of the unrooted

Fig. 3. (a) Two replug moves transform a tree T_1 into another tree T_2. Replug moves modify one end of an edge, like SPR moves, but may create loops and disconnected components. The dashed line with an arrow is the edge that will be added in the next step; simultaneously the dotted edge touching the tail end of the arrow will be removed. (b) A maximum endpoint agreement forest (MEAF) of T_1 and T_2. The ϕ nodes indicate endpoints of removed edges that are common to both trees. In this case, as there are two components and two ϕ-nodes, the TBR and replug distances are both 2. The SPR distance is 3. (c) A MEAF of the trees from Fig. 2a. In this case, the replug distance and SPR distance are both 3, while the TBR distance is 2.

SPR distance in Section 3 which make it difficult to solve using agreement forests. Property 3.1 states that one or both ends of an edge may move in an optimal unrooted SPR path. Recall that each SPR operation moves one endpoint of an edge and so any agreement forest formulation must be able to represent which side of an edge remains fixed, as well as the case where both sides of an edge must be moved. The replug distance also has this directional property. However, as replug moves are not required to maintain a tree structure, they do not prevent other replug moves in the way that uSPR moves do. We thus have the following observation:

Observation 5.2. Given an SAF, an endpoint of any connection can be attached to any socket with a valid replug move.

We can use this observation to show that the replug distance does not have Properties 3.2, 3.3 and 3.4 of the unrooted SPR distance.

Lemma 5.3. Let $M = m_1, m_2, \ldots, m_k$ be an optimal sequence of replug moves transforming a tree T_1 into a tree T_2. Then M does not (1) move the same endpoint u of an edge twice, (2) break a common cluster of T_1 and T_2, or (3) break an edge of a common path of T_1 and T_2 with respect to M.

To account for Property 3.1 and define an agreement forest that represents replug operations, we thus need to represent a fixed endpoint of a moved edge (analogous to component roots of a rooted AF) as well as the case where both endpoints are moved (analogous to an unrooted AF for TBR calculation). We will call this augmented tree structure a (phylogenetic) X-ϕ-tree, which is a generalization of an X-tree. It has $n \geq |X|$ labeled leaves, $|X|$ of which are uniquely labeled from X, and the remaining $q(T) := n - |X|$ leaves are each labeled ϕ. These ϕ nodes will be used to indicate an edge endpoint which remains fixed during a set of

tree moves. As with X-trees, two X-ϕ-trees are considered equal if and only if there is an isomorphism between their nodes and edges that maintains node labels. Thus, ϕ nodes are interchangeable. An X-ϕ-forest is a forest of X-ϕ-trees.

Consider the three possiblities of an edge $e = (u, v)$ of an X-ϕ-tree that is modified by one or more replug moves:

(1) the v endpoint is moved such that e becomes (u, x)
(2) the u endpoint is moved such that e becomes (w, v)
(3) both endpoints are moved such that e becomes (y, z).

In order to describe such changes as part of a type of agreement forest, we attach ϕ nodes as follows.

In the first case we represent this replug operation by cutting edge e, suppressing node v, then attaching a new leaf labeled ϕ to node u. The second case is similar: we suppress u then add a ϕ node to v. Observe that the third case is equivalent to two replug operations or a single TBR operation. We can thus represent the third case by cutting edge e and suppressing nodes u and v, as in general unrooted AFs.

To complement the notion of "directional" agreement forest we also need a notion of directional edge set, as follows. We define an *endpoint edge* E_c of a tree T to be an edge along with a proper subset of its endpoints. This will be denoted $e \rhd p$ for an edge $e = (u, v)$ where $p \subsetneq \{u, v\}$. Note that $p = \emptyset$ indicates that both endpoints are moved. In the context of a sequence of replug moves, these subsets p are the nodes that remain fixed, which we will call *augmented endpoints*. An endpoint edge of an X-ϕ-tree cannot have a ϕ node as an augmented endpoint.

An endpoint edge set E is a set of endpoint edges E_c of T. For an endpoint edge set E, we use $F - E$ to denote the X-ϕ-forest obtained by deleting the edges in E from F and adding ϕ node neighbors to each augmented endpoint. We call this "cutting" the endpoint edge set, which is in general a many-to-one mapping. $F \div E$ is again the (X-ϕ) forest yielded by $F - E$ and we say that $F \div E$ is an endpoint forest of F.

An *endpoint agreement forest* is now naturally defined in terms of endpoint forests. Given trees T_1 and T_2 and forests F_1 of T_1 and F_2 of T_2, a forest F is an EAF of F_1 and F_2 if it is an endpoint forest of both trees. Observe that an EAF is a generalization of an AF, where either no nodes remain fixed (unrooted AF, in which every $p = \emptyset$), or every endpoint furthest from the root remains fixed (rooted AF, in which no $p = \emptyset$). As such, we can always find an EAF for two trees by constructing an AF.

The *weight* of an EAF F is defined as:

$$\omega(F) = 2(|F| - 1) - q(F),$$

where $q(F)$ is the number of ϕ nodes in F. Observe that the weight strictly increases upon cutting an endpoint edge set.

We say that an EAF F of two trees T_1 and T_2 is a *maximum endpoint agreement forest* (MEAF) of T_1 and T_2 if it has minimum weight (Figs. 3b and 3c). Use $\omega(T_1, T_2)$ to denote this minimum weight. We can show that $d_R(T_1, T_2)$ is equal to $\omega(T_1, T_2)$, rather than the number of components.

Theorem 5.4. Let T_1 and T_2 be unrooted trees. Then $d_R(T_1, T_2) = \omega(T_1, T_2)$.

Our proof of Theorem 5.4 provides an inductive procedure for constructing an optimal sequence of replug moves from an MEAF (Fig. S1 in the appendix, available in the online supplemental material). Moreover, each step of this procedure can be implemented to require linear time using the tree to AF mappings we construct later with Lemma 6.3. Thus:

Corollary 5.5. *Let F be an EAF of two unrooted trees T_1 and T_2. A sequence of $\omega(F)$ replug moves that transform T_1 into T_2 can be obtained from F in $O(n\omega(F))$-time.*

The replug distance is a lower bound for the SPR distance, which enables the fast SPR distance algorithm in Section 7.

Theorem 5.6. *For any pair of trees T_1 and T_2,*

$$d_{\mathrm{TBR}}(T_1, T_2) \leq d_{\mathrm{R}}(T_1, T_2) \leq d_{\mathrm{SPR}}(T_1, T_2).$$

6 A FIXED-PARAMETER REPLUG DISTANCE ALGORITHM

In this section we develop a fixed-parameter algorithm for computing the replug distance between a pair of binary rooted trees T_1 and T_2. We apply a two-phase bounded search tree approach to determine whether the replug distance is at most a given value k. The minimum replug distance can be found by repeatedly running this algorithm with increasing parameters $k = 1, 2, \ldots, d_{\mathrm{R}}(T_1, T_2)$.

Our strategy is to enumerate all possible maximal agreement forests (Section 6.1) and then decorate them with ϕ nodes to enumerate the possible maximal endpoint agreement forests (Section 6.2). We then improve the second phase by considering each possible socket assignment as a SAF of F and finding an optimal ϕ node assignment (Section 6.3).

6.1 Enumerate Maximal AFs (TBR Distance)

The first phase of our search enumerates all maximal (unrooted) AFs. We modify the $O(4^k n)$-time MAF (TBR) distance algorithm of Whidden and Zeh [54] to do so, and adopt some of the improved branching cases from the $O(3^k n)$-time algorithm of Chen et al. [55]. Neither algorithm is capable of enumerating all mAFs in its original form and, furthermore, we can not apply all of the cases from Chen et al. because some of their cases necessarily miss some mAFs.

Given two trees T_1 and T_2 and a parameter k, our algorithm $\mathsf{mAF}(T_1, T_2, k)$ finds each maximal AF that can be obtained by cutting k or fewer edges. If no such mAF exists, then it returns an empty set. If mAFs are found, we apply the next phase of our search to each of these candidate underlying AFs. If we find no mAFs or if none of these mAFs underly a maximum endpoint agreement forest, we increase k and repeat until we find one. This does not increase the running time of the algorithm by more than a constant factor because the running time depends exponentially on k.

Our algorithm is recursive, acting on the current forests of T_1 and T_2 that are obtained by cutting edge sets E_1 and E_2, respectively, with $|E_1| = |E_2|$. These forests maintain a specific structure, $F_1 = T_1 \div E_1 = \{\dot{T}_1\} \cup F_0$, and $F_2 = T_2 \div E_2 = \dot{F}_2 \cup F_0$, defined as follows:

- The tree $\dot{T}_1$ is obtained from T_1 by cutting E_1.
- The forest F_0 has all of the rooted subtrees cut off from T_1. The rooting for these subtrees is that given by rooting at the node touching a cut edge.

- The forest $\dot{F}_2$ has the same label set as $\dot{T}_1$, and contains the components of F_2 that do not yet agree with $\dot{T}_1$ (i.e., $(F_2 \div E_2) \setminus F_0$).
- A set R_t (roots-todo) stores the roots of (not necessarily maximal) subtrees agreeing between $\dot{T}_1$ and $\dot{F}_2$.

Each invocation of the algorithm works to modify $\dot{T}_1$ and $\dot{F}_2$ towards agreement, collecting the needed agreement forest components in F_0. For the top-level invocation, $\dot{T}_1 = T_1$, $\dot{F}_2 = \{T_2\}$, $F_0 = \emptyset$, and R_t contains all leaves of T_1.

We say that a pair of nodes $a, c \in R_t$ that are siblings in $\dot{T}_1$ are a *sibling pair* $\{a, c\}$. Such a pair must exist when $|R_t| \geq 2$ just as every tree must have a pair of sibling leaves. For each node $x \in R_t$, e_x denotes the unique edge of $\dot{T}_1$ or $\dot{F}_2$ that is adjacent to x and extends out of the common subtree rooted at x. We call this the *edge adjacent to x*.

Algorithm. $\mathsf{mAF}(F_1, F_2, k)$

1. (Failure) If $k < 0$, then return $\emptyset$.
2. (Success) If $|R_t| = 1$, then $F_1 = F_2$. Hence, F_2 is an AF of T_1 and T_2. Return $\{F_2\}$.
3. (Prune maximal agreeing subtrees) If there is no node $r \in R_t$ that is a root in $\dot{F}_2$, proceed to Step 4. Otherwise choose such a node $r \in R_t$; remove it from R_t and move the subtree S of $\dot{F}_2$ containing r to F_0; cut the edge e_r separating S in $\dot{T}_1$ from the rest of $\dot{T}_1$, which gives S a rooting. Return to Step 2.
4. Choose a sibling pair $\{a, c\}$ of $\dot{T}_1$. Let e_a and e_c be the edges adjacent to a and c in $\dot{F}_2$.
5. (Grow agreeing subtrees) If a and c are siblings in $\dot{F}_2$, do the following: remove a and c from R_t; label their mutual neighbor in both forests with (a, c) and add it to R_t; return to Step 2.
6. (Branching) Distinguish two cases depending on whether $a \sim_{F_2} c$ (Fig. 4). Note that because a and c are in $\dot{T}_1$ they are not in F_0, so $a \sim_{F_2} c$ is equivalent to $a \sim_{\dot{F}_2} c$.

   6.1. If $a \not\sim_{F_2} c$, try cutting off either the subtree rooted at a or that rooted at c, returning:

   $$\mathsf{mAF}(F_1, F_2 \div \{e_a\}, k - 1) \cup \mathsf{mAF}(F_1, F_2 \div \{e_c\}, k - 1).$$

   6.2. If $a \sim_{F_2} c$, let $b_1, b_2, \ldots, b_q$ be the pendant nodes on the path from a to c in T_2 and let e_{b_i} be the pendant edge adjacent to b_i. Try cutting e_a or e_c, or try cutting off all but one of the subtrees on the path between a and c, returning:

   $$\mathsf{mAF}(F_1, F_2 \div \{e_a\}, k - 1)$$
   $$\cup \mathsf{mAF}(F_1, F_2 \div \{e_{b_2}, e_{b_3}, \ldots, e_{b_q}\}, k - (q - 1))$$
   $$\cup \mathsf{mAF}(F_1, F_2 \div \{e_{b_1}, e_{b_3}, e_{b_4}, \ldots, e_{b_q}\}, k - (q - 1))$$
   $$\cdots$$
   $$\cup \mathsf{mAF}(F_1, F_2 \div \{e_{b_1}, e_{b_2}, \ldots, e_{b_{q-1}}\}, k - (q - 1))$$
   $$\cup \mathsf{mAF}(F_1, F_2 \div \{e_c\}, k - 1).$$

The correctness of our algorithm is proven using a simplification of the proof of Theorem 2 of Whidden and Zeh [54], modified for arbitrary maximal agreement forests.

Lemma 6.1. $\mathsf{mAF}(T_1, T_2, k)$ *returns the set of maximal agreement forests of T_1 and T_2 that can be obtained by cutting k or fewer edges.*
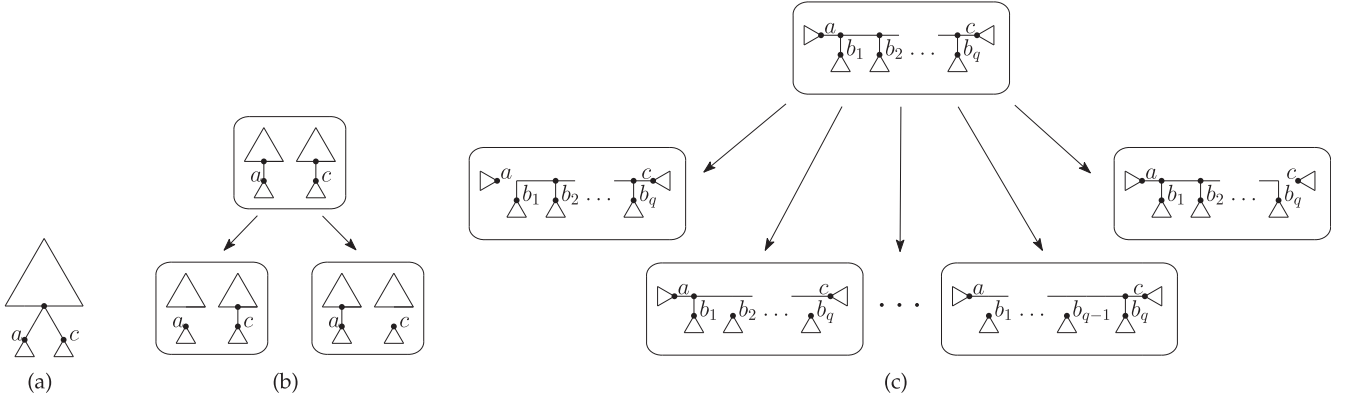
Fig. 4. Branching cases of the maximal AF enumeration algorithm. (a) A sibling pair $\{a, c\}$ of F_1. (b) The branching cases applied to F_2 when $a \not\sim_{F_2} c$. (c) The branching cases applied to F_2 when $a \sim_{F_2} c$.

Theorem 6.2. *Given two trees T_1 and T_2, the set S of maximal agreement forests of T_1 and T_2:*

(1) *can be enumerated in $O(4^k n)$-time,*

(2) *can be returned in $O(4^k n + |S| \log |S|)$-time (the $\log |S|$ factor required to avoid duplicate agreement forests), and*

(3) *is of size at most 4^k.*

6.2 Enumerate Maximal EAFs (Replug Distance)

The second phase of our algorithm enumerates maximal endpoint agreement forests to compute the replug distance. Using $\mathsf{mAF}(T_1, T_2, k)$ we can enumerate each mAF F of two trees T_1 and T_2 that has $k + 1$ or fewer components, in other words, the mAFs that correspond to k or fewer TBR operations. However, such an mAF may imply anywhere from k to $2k$ SPR operations. We assign ϕ nodes to F to determine the maximal EAF (mEAF) with minimum weight.

We require a method to map nodes of the trees to the agreement forest and vice versa. Given an mAF F of trees T_1 and T_2, we construct a "forward" mapping $\psi(x)$ that maps nodes of T_1 and T_2 to nodes of F, as well as a "reverse" mapping $\psi^{-1}(T_i, x), i \in \{1, 2\}$, that maps nodes of F to the trees. Recall that $F = T_1 \div E_1 = T_2 \div E_2$, where E_1 and E_2 are the edges cut from T_1 and T_2, respectively. As F contains fewer nodes than T_1 and T_2, our forward mapping $\psi(x)$ maps some nodes of T_1 and T_2 to the empty set. These are exactly the nodes that are contracted when E_1 and E_2 are cut to obtain F from T_1 and T_2. We call these the *dead* nodes of the trees, in contrast to the *alive* nodes that are mapped to nodes of F.

There are potentially many edge sets that can be cut from the trees to obtain the mAF F, but we can quickly obtain one pair of such edge sets and their induced mappings:

Lemma 6.3. *Let F be an mAF of two trees T_1 and T_2. Then*

(1) *two sets of edges E_1 and E_2 such that $T_1 \div E_1 = T_2 \div E_2 = F$ can be constructed in linear time, and*

(2) *a mapping $\psi(x)$ from T_1 and T_2 nodes to nodes of F and the reverse mappings $\psi^{-1}(T_i, x)$ can be built in $O(n)$-time; lookups using these mappings take constant time.*

Now, we develop an algorithm $\mathsf{replug}(T_1, T_2, F, d)$ to enumerate the set of mEAFs of weight d or less with the same components (modulo ϕ nodes) as a given mAF F of trees T_1 and T_2. First assume that the edge set E_1 to remove from T_1

is pre-specified. In this case, each corresponding mEAF F' is simply F with a choice of endpoint for each edge $e = (u, v) \in E_1$, that is, $e \triangleright \{u\}$, $e \triangleright \{v\}$, or $e \triangleright \emptyset$. We have three choices for each edge, and can therefore enumerate these candidate mEAFs in $O(3^k n)$ time, where $k + 1$ is the number of components of F, and keep any of weight d or less. Note that, for each candidate mEAF, we must also check, using $O(n)$-time, whether it is a forest of T_2 (it is guaranteed to be a forest of T_1 as we enumerated choices of endpoints for E_1). We define the following recursive subprocedure to enumerate the candidate mEAFs that can be obtained from an mAF F with a given set of edges E_1.

Algorithm. replug-decorate(T_1, T_2, F, E_1, k)

1. (Failure) If $k < 0$ then return $\emptyset$.
2. (Success) If F contains an unprocessed edge $e = (u, v) \in E_1$, then proceed. Otherwise F is a forest of T_1 with weight at most double the initial value of k. If F is also a forest of T_2, return $\{F_2\}$. Otherwise, return $\emptyset$.
3. (Branch) Return:
   replug-decorate$(T_1, T_2, F \div e \triangleright \{u\}, E_1 \setminus \{e\}, k - 1) \cup$
   replug-decorate$(T_1, T_2, F \div e \triangleright \{v\}, E_1 \setminus \{e\}, k - 1) \cup$
   replug-decorate$(T_1, T_2, F \div e \triangleright \emptyset, E_1 \setminus \{e\}, k - 1)$.

Define a *dead tree* to be a tree induced by some maximal set of adjacent edges in T_1 or T_2 that do not map to edges of F. When every dead tree is a single edge, the edge sets E_1 and E_2 removed to obtain F from T_1 and T_2 are unique and thus known. However, in general we must enumerate every set of edges E_1 such that $T_1 \div E_1 = F$, because a component C of F that is adjacent to a dead tree with three or more leaves in both T_1 and T_2 may have nontrivial ϕ node structures in the mEAF. Given the node mappings and reverse node mappings constructed with Lemma 6.3, we can identify the dead trees in a tree with three preorder traversals. The node mapping construction procedure roots the trees arbitrarily. We use the same roots in our traversals.

The first traversal determines the depth from each node of T_1 to its arbitrary root. The second traversal identifies the path induced by each edge $e = (u, v)$ of F in T_1. To do so, we move towards the root from $\psi^{-1}(T_1, u)$ and $\psi^{-1}(T_1, v)$, always moving from the more distant node. These paths will cross at the least common ancestor of $\psi^{-1}(T_1, u)$ and $\psi^{-1}(T_1, v)$ with respect to the root. We mark $\psi^{-1}(T_1, u)$ and $\psi^{-1}(T_1, v)$ as alive and each other node of the path as a

socket. If u or v is a component root of F, then it was not removed by a forced contraction when its adjacent edge was cut. In this case we also mark $\psi^{-1}(T_1, u)$ or $\psi^{-1}(T_1, v)$, respectively, as a socket. The third and final traversal marks all the remaining nodes as dead and identifies the connected components of dead trees. We will use this assignment of dead nodes and socket nodes at the end of this section to quickly find a single optimal ϕ node assignment.

A dead tree with q leaves has $2q - 3$ edges. Each such tree can be removed by removing $q - 1$ edges and contracting the resulting degree two nodes. Thus there are $\binom{2q-3}{q-1} \leq 4^q$ choices of edges to cut that result in the same dead tree. This implies that there are at most 4^k edge sets that result in the same AF. We thereby construct each possible E_1 set and test each combination of three choices per edge, as in the unique edge set case. This requires $O(4^k 3^k n) = O(12^k n)$ time.

In summary, the high-level steps of the algorithm to calculate the replug distance are as follows.

Algorithm. replug(T_1, T_2, F, d)

---

1. Root T_1 and T_2 arbitrarily at nodes r_1 and r_2.
2. Construct the mappings ψ and ψ^{-1} using Lemma 6.3.
3. Compute the distance from each node n in T_1 to r_1.
4. For each edge $e = (u, v)$ of F, mark $\psi^{-1}(T_1, u)$ and $\psi^{-1}(T_1, v)$ as alive and the other nodes on the path in T_1 from $\psi^{-1}(T_1, u)$ to $\psi^{-1}(T_1, v)$ as sockets.
5. Mark all unmarked nodes dead.
6. Identify the dead components $D_1 = \{d_1, d_2, \ldots d_{q_1}\}$ of T_1.
7. Repeat the previous steps for T_2, identifying the dead components D_2.
8. Let $\mathcal{F} \leftarrow \emptyset$.
9. For each edge set E_1 induced by D_1,

   a) Let $k \leftarrow |E_1|$.
   b) Let $\mathcal{F}' \leftarrow$ replug-decorate(T_1, T_2, F, E_1, k).
   c) For each forest $F' \in \mathcal{F}'$, if F' has weight at most d and is consistent with some edge set E_2 induced by D_2 then add it to $\mathcal{F}$.
10. Return $\mathcal{F}$.

In combination with the algorithm mAF(T_1, T_2, k) from Section 6.1, this implies:

Theorem 6.4. *Given two trees T_1 and T_2, the set of mEAFs of T_1 and T_2 with $k + 1$ or fewer components can be enumerated in $O(48^k n)$-time and there are at most 48^k such mEAFs.*

This is a fairly loose bound as there are typically far fewer than 4^k mAFs [56] and the majority of agreement forests will have few and small dead trees. Although the above procedure requires more effort to enumerate the mEAFs that can be obtained from a single mAF in the worst case than it does to enumerate each of the mAFs, we expect the opposite to be the typical case in practice because closely related trees (such as those compared in phylogenetic analysis) rarely differ in only a single large connected set of bipartitions in both trees which is necessary to induce a single large dead component in both trees.

We can improve the ϕ node assignment step in the case that k is close to the SPR distance. If the SPR distance is d then it never makes sense to consider an mEAF with weight

exceeding d for the final phase of our search. In fact, we can exclude any mEAF F' with fewer than $d - k\, \phi$ nodes, as their weight (and thus replug distance) is guaranteed to exceed d. We thus define replug-decorate*(T_1, T_2, F', E_1, d), a version of replug-decorate which takes this observation into account to enumerate a more limited set of candidate mEAFs. For the definition, initially $F' = T_1$ and E_1 contains each edge of F, as follows:

Algorithm. replug-decorate*(T_1, T_2, F', E_1, d)

---

1. (Failure) If $d < 0$ then return $\emptyset$.
2. (Success) If F' contains an unprocessed edge $e = (u, v) \in E_1$, then proceed. Otherwise F' is a forest of T_1 with weight less than the initial value of d. If F' is also a forest of T_2, return $\{F_2\}$. Otherwise, return $\emptyset$.
3. (Branch) Return:
   replug-decorate*$(T_1, T_2, F' \div e \triangleright \{u\}, E_1 \setminus \{e\}, d - 1) \cup$
   replug-decorate*$(T_1, T_2, F' \div e \triangleright \{v\}, E_1 \setminus \{e\}, d - 1) \cup$
   replug-decorate*$(T_1, T_2, F' \div e \triangleright \emptyset, E_1 \setminus \{e\}, d - 2)$.

Theorem 6.5. *Given two trees T_1 and T_2, the set of mEAFs of T_1 and T_2 with weight at most d and $k + 1$ or fewer components can be enumerated in $O(\min(4^k, 2.42^d) \cdot 12^k n)$-time and there are at most $(\min(4^k, 2.42^d) \cdot 12^k)$ such mEAFs.*

6.3 Quickly Finding a Single MEAF (Replug Distance)

Finally, we discuss how to quickly compute the replug distance in practice. Our goal now is to determine whether the replug distance between two trees is at most d. The minimum distance can again be found by testing increasing values of d starting from 0. We again enumerate the maximal agreement forests that can be obtained by removing at most d edges with mAF(T_1, T_2, d). For each such mAF F, however, we now wish to find a single optimal ϕ node assignment, rather than each such optimal assignment.

To determine such an optimal assignment, we again start by finding an initial set of edges E_1 and E_2 such that $T_1 \div E_1 = T_1 \div E_2 = F$. Our first optimization stems from the observation that nontrivial ϕ node structures are only possible when a component C of an mAF is adjacent to a dead tree in both trees T_1 and T_2. Call a dead tree *uncertain* if it is adjacent to such a component in a given tree. When enumerating edge sets E_1' and E_2' that can be used to obtain F, we only need to enumerate combinations of dead tree edges from dead trees that are uncertain in both T_1 and T_2.

Now, given an AF F of T_1 and T_2 and edge sets E_1 and E_2 that can be used to obtain F from the trees, we develop our optimal ϕ node assignment procedure. The intermediate structures are illustrated in Fig. S2 in the appendix, available in the online supplemental material. We first identify the set of candidate agreement forest edges that can be adjacent to a ϕ-node (*candidate ϕ-nodes*). We next identify the set of constraints on EAFs induced by these candidates and show that such sets belong to the monotone class $\mathrm{CNF}_+(\leq 2)$ [57]. This is the class of satisfiability formulas in conjunctive normal form with no negated literals in which each literal occurs in at most 2 clauses. As such, we reduce the problem of decorating an AF with $d - k\, \phi$ nodes, to that of determining whether a boolean $\mathrm{CNF}_+(\leq 2)$ formula can

be satisfied by an assignment with at least $d - k$ variables set to true. Finally, we briefly describe how this latter problem can be solved in $O(k^{1.5})$-time by finding a minimum edge cover in an equivalent clause graph [57].

To identify the candidate ϕ-nodes, we convert the AF F to an SAF S. Recall that a ϕ-node indicates an edge endpoint which remains fixed during a set of moves transforming one tree, T_1, into another, T_2. This implies that one socket remains fixed for each ϕ-node decorated edge in the T_1 and T_2 configurations of S. Thus, the sockets of S which are connected to an edge in both T_1 and T_2 are exactly our set of candidate ϕ-nodes.

To accomplish this SAF conversion, we must thus introduce sockets from both T_1 and T_2 and match sockets which represent nodes from both trees. As described in the replug algorithm, a path of dead nodes in T_1 (respectively, T_2) between two alive nodes form a set of T_1 (T_2) sockets. We say that these dead nodes are all on the same edge of F. These sets are ambiguous in the sense that any pair consisting of a T_1 and T_2 socket that are on the same edge may be a fixed endpoint that can have a ϕ node. We can identify these ambiguous sets, as well as the dead nodes which are not sockets, in linear time as previously discussed. We must choose a mapping for each such set of h_1 T_1 sockets and h_2 T_2 sockets that map to the same alive node edge and that maintains the same orientation. There are thus $\binom{h_1}{h_2}$ choices for each such set (assuming $h_1 \geq h_2$), for a maximum of at most $4^{h_1} \leq 4^k$ combinations that must be considered. We further observe that dead tree nodes are not possible choices in such a mapping. Therefore, there are at most 4^k SAFs that must be considered, stemming from both choices of E_1 and E_2 and socket mappings. We expect such situations to be degenerate in practice and that most cases will induce a small number of SAFs based on our experience with rooted agreement forests.

Now, for each SAF, we identify the set of constraints induced by the candidate ϕ-nodes. These constraints naturally arise from the fact that we cannot assign two ϕ-nodes to both endpoints of the same cut edge in E_1 of T_1 or E_2 of T_2, as that would imply that the edge remains fixed. Now, suppose, without lack of generality, we have a dead tree D adjacent to n_D sockets of F in T_1. We say that an assignment of ϕ-nodes to sockets of S *satisfies* D if at least one of the n_D sockets adjacent to D is not assigned a ϕ node. By Lemma 6.6, it suffices to find an assignment of ϕ nodes to S that satisfy each dead component of T_1 and T_2.

Lemma 6.6. *Given a socket agreement forest S of two trees T_1 and T_2, an assignment of ϕ-nodes to sockets of S is an EAF of T_1 and T_2 if, and only if, the assignment satisfies every dead tree in T_1 and T_2 with respect to S.*

We now show that the full set of such constraints is a boolean monotone $\mathrm{CNF}_+(\leq 2)$ formula [57]. In particular, determining the maximum number of ϕ nodes that can be added to an SAF is thus equivalent to determining the minimum number of variables which must be true in such a formula—the minimum cardinality satisfiability problem.

Lemma 6.7. *The replug distance problem on an SAF S of trees T_1 and T_2 can be solved by solving the minimum cardinality satisfiability problem on a boolean monotone $\mathrm{CNF}_+(\leq 2)$ formula.*

This satisfiability problem can be solved efficiently with a polynomial time algorithm. Unlike general CNF satisfiability, which is NP-hard [57], the minimum cardinality satisfiability problem on monotone $\mathrm{CNF}_+(\leq 2)$ formulas is equivalent to the edge covering problem on a *clause graph*.

Lemma 6.8. *Given an SAF F of two trees T_1 and T_2, an EAF F' of T_1 and T_2 can be computed in $O(k^{1.5})$-time such that F' has F as its underlying agreement forest and F' contains as many ϕ-nodes as any such EAF.*

Finally, we combine the fast ϕ-node assignment procedure of Lemma 6.8 with our general mAF enumeration to solve the replug distance for a pair of trees:

Theorem 6.9. *Given two trees T_1 and T_2, an EAF F of T_1 and T_2 with $\omega(F) = d_{\mathrm{R}}(T_1, T_2)$ and $k + 1$ components can be found (or determined not to exist) in:*

    (1) $O(4^k(4^k k^{1.5} + n))$-time, or

    (2) $O(4^k n + Y k^{1.5})$-time, where Y is the number of candidate SAFs with at most $k + 1$ components.

Repeated applications of the subtree and chain reduction rules reduce the size of the initial trees to a linear function with respect to their SPR distance [50]. We thus achieve:

Corollary 6.10. *The replug distance for a pair of trees T_1 and T_2 can be solved in $O(16^d d^{1.5})$-time, where $d = d_{\mathrm{R}}(T_1, T_2)$.*

7 COMPUTING THE uSPR DISTANCE WITH A PROGRESSIVE A* APPROACH

We now present our fixed-parameter algorithm for computing the subtree prune-and-regraft distance between two unrooted trees, T_1 and T_2. We first reduce the trees repeatedly by applying the subtree and chain reduction rules. After these reductions, the trees contain at most $28 d_{\mathrm{SPR}}(T_1, T_2)$ leaves [50]. We then apply an incremental heuristic A* search [58] beginning from T_1 to find T_2. We could simply use standard A* with a replug distance heuristic. However, the replug distance is relatively expensive to compute, and such an algorithm would be very slow. Instead, we use a sequence of increasingly more accurate but expensive to compute heuristics in a method we call *progressive A* search*. The benefit of our algorithm over a standard A* search lies in the use of multiple heuristic functions, each of which provides a lower bound on its successor and is significantly less expensive to compute. This search focuses on paths that likely lead to the target tree while avoiding expensive computation of the TBR and replug distances on each of the $O(n^2)$ neighbors of every visited tree.

We maintain a priority queue P of trees that remain to be explored. We also maintain a set V of trees that have already been visited, along with their estimated distances to T_2 and what function was used to make that estimate. Initially $P = \{T_1\}$. Define the priority $p(T) = (d_T, h_T, e_T)$. First, d_T is the distance already traveled: $d_T = d_{\mathrm{SPR}}(T_1, T)$. Second, h_T is the estimated minimum distance from T_1 to T_2 that can be achieved by a path that visits tree T. This estimate h_T is $d_T + e_T(T, T_2)$, where $e_T(T, T_2)$ is the estimated minimum distance from T to T_2 using the estimation function e_T. Third, e_T is one of the four estimation functions (i.e., the heuristics) used as follows. The heuristic $\mathrm{ONE}(T, T_2) = 1$ for

any tree. The heuristic $a_{\text{TBR}}(T, T_2)$ is the linear-time 3-approximation algorithm for the TBR distance of Whidden and Zeh [54] divided by 3 to guarantee a lower bound estimate of the TBR distance. $d_{\text{TBR}}(T, T_2)$ and $d_{\text{R}}(T, T_2)$ are computed with the algorithms in the preceding section. We impose a total order on these heuristics as described below.

Our search procedure always considers the next tree from P with smallest priority according to the partial ordering of these values where for any two trees T_i and T_j, $p(T_i) < p(T_j)$ iff:

(1) $h_{T_i} < h_{T_j}$,
(2) $h_{T_i} = h_{T_j}$ and $e_{T_i} < e_{T_j}$, or
(3) $h_{T_i} = h_{T_j}$, $e_{T_i} = e_{T_j}$, and $d_{T_i} > d_{T_j}$.

In other words, we prioritize the tree T_i with smallest heuristic distance h_{T_i}. We break h_T ties using a total ordering of our heuristics e_T: $\text{ONE}() < a_{\text{TBR}}() < d_{\text{TBR}}() < d_{\text{R}}()$. In turn, e_T ties are broken by partial distances d_T.

Each estimator provides a lower bound on each of the successive estimators as well as the target distance $d_{\text{SPR}}(T_i, T_2)$, which is an important condition to ensure the correctness of our progressive A* search. We break ties by selecting the tree that is most distant from the starting position and therefore estimated to be closer to the destination tree. Trees with equal h_T, e_T, and d_T values are selected from uniformly at random.

Initially, $p(T_1) = (0, 1, \text{ONE}())$ when inserting T_1 into P. We repeatedly remove the tree T from P with smallest priority $p(T) = (d_T, h_T, e_T)$ and apply one of the options:

(1)   if $e_T = \text{ONE}()$, reinsert T with priority $(d_T, d_T + a_{\text{TBR}}(T, T_2), a_{\text{TBR}}())$.
(2)   if $e_T = a_{\text{TBR}}()$, reinsert T with priority $(d_T, d_T + d_{\text{TBR}}(T, T_2), d_{\text{TBR}}())$.
(3)   if $e_T = d_{\text{TBR}}()$, reinsert T with priority $(d_T, d_T + d_{\text{R}}(T, T_2), d_{\text{R}}())$.
(4)   if $e_T = d_{\text{R}}()$, explore each of the $\text{O}(n^2)$ trees that can be obtained from T by one SPR operation and insert each such tree $t \notin V$ into P with priority $(d_T + 1, 1, \text{ONE}())$ and into V. However, if any of T's SPR neighbors are T_2 then we terminate the program and return the SPR distance of $d_T + 1$.

Theorem 7.1. *The SPR distance between two unrooted trees T_1 and T_2 can be computed in $\text{O}(Y\,16^d d^{1.5})$ time, where Y is the number of trees explored by the heuristic, and $d = d_{\text{SPR}}(T_1, T_2)$. Note that $Y = \text{O}((28d)!!)$ after reducing T_1 and T_2.*

8   EXPERIMENTAL EVALUATION

We implemented our algorithms in the C++ program uspr [59]. Given two unrooted trees, this software can compute their TBR 3-approximation-based lower bound, TBR distance, replug distance, or SPR distance. This program was tested on the prokaryote dataset of [8] which compares a phylogenetic supertree constructed by the Matrix Representation with Parsimony method to 22,437 individual gene trees ranging from 4–144 taxa. This dataset has been widely used to test methods for computing rooted SPR distances [21], [36], [54], as well as the sole previous software for computing uSPR distances sprdist by Hickey et al. [35]. (Note that this software is not the identically named software by different authors for computing rooted SPR distances [41], [60].) In the remainder of
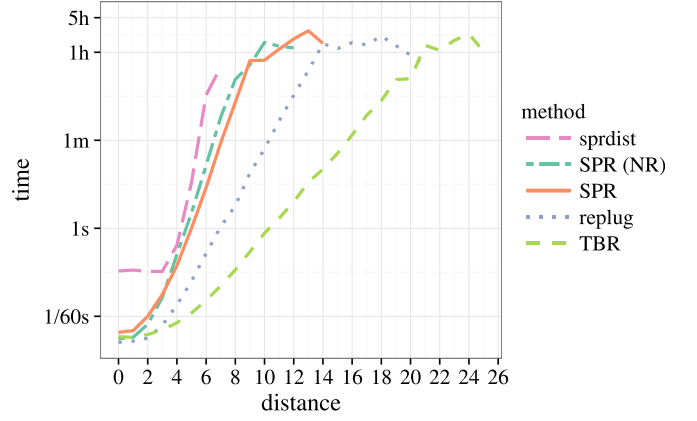

Fig. 5. The mean time on a log-60 scale required as a function of distance to compute SPR distances using sprdist and SPR, replug, and TBR distances using uspr. The SPR test used all four bounding heuristics. The SPR (NR) test used all the bounding heuristics except the replug distance. Means were computed only over computations that succeeded for a given run within the time and memory limits.

this paper we refer only to the software of Hickey et al. as sprdist.

We computed SPR distances using uspr and sprdist on a computer cluster running Ubuntu 14.04 with the SLURM cluster management software. We allocated one Intel Xeon X5650 CPU per computation and terminated instances which required more than 4,096 MB of memory or 5 hours. We tested the 5,689 gene trees with 10 or more taxa. We used the standard practice of comparing each gene tree to the subset of the supertree with identical taxa only.

8.1   Running Time

Our new algorithms allow us to compute much larger SPR distances than were previously possible, as well as compute the same distances with much less time and memory (Fig. 5). Note that Fig. 5 shows the mean time required by all computations that completed successfully given the time and memory limit, and summarizes different numbers of computations for different methods. We computed SPR distances as large as 14 with uspr, double the maximum distance of 7 computable with sprdist. sprdist also uses a graph exploration strategy, but without efficient heuristics to guide the search it typically reached the memory limit before the time limit was reached. In contrast, uspr explores fewer trees but spends more time per tree and is therefore CPU bound and more scalable than sprdist. Moreover, uspr found 176 instances with an SPR distance of 7 with a mean time of 53.29 seconds, nearly two orders of magnitude faster than the 33 instances with an SPR distance of 7 computed by sprdist with a mean time of 1,808.96 seconds.

We tested our software with and without the replug distance heuristic to determine if the better distance estimate outweighed the extra computation required to compute the heuristic. The replug distance heuristic was necessary to compute SPR distances of 13 or 14. Our software was an average of 5x faster with the replug heuristic, taking a mean time of 20.59 s compared to 98.62 s with only the TBR heuristic on problem instances which both methods completed given the running time limit. The replug heuristic greatly reduced the number of trees examined, and therefore the memory required, by a factor of about 33 to a mean of
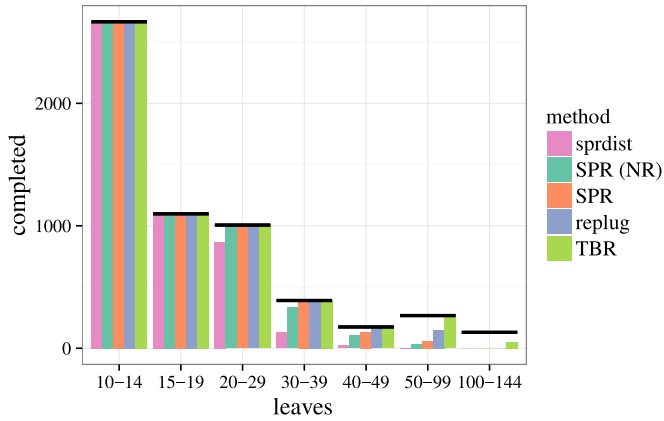
Fig. 6. The number of successfully computed SPR distances using `sprdist` and SPR, replug, and TBR distances using `uspr` given a time limit of five hours and memory limit of 4,096 MB. Results are summarized by ranges of the number of leaves in the tree pairs, with the black lines indicating the total number of tree pairs in the given range.

Fig. 7. Mean SPR, replug, and TBR distances of all tree pairs for which the SPR distance was succesfully computed using `uspr` given a time limit of five hours and memory limit of 4,096 MB.

2,921.4 with the heuristic compared to 102,498.8 without the heuristic when each approach completed.

As expected, our methods can be used to compute much larger replug distances (at most 21) and TBR distances (up to 25) given 5 hours. TBR distance computations are orders of magnitude faster than replug distance computations. As both distances are used repeatedly during our SPR distance progressive A* search, the search time depends primarily on the time required to compute these distances as well as the number of such distances that must be computed.

8.2 Completion
Larger trees may have larger distances from a reference tree or supertree so we summarized the number of successfully computed distances for defined ranges of tree sizes (Fig. 6). Our new SPR distance algorithm is practical for trees with up to 50 leaves. We successfully computed distances for 132 of the 170 tree pairs with 40-49 leaves and 5,142/5,151 of the tree pairs with fewer than 40 leaves. However, we were only able to compute SPR distances for 60 of the 261 tree pairs with 50-99 leaves. This is in stark contrast to `sprdist` which can not reliably handle trees with more than 30 leaves, as it could only compute distances for 867 of the 1,004 pairs of trees with 20-29 leaves, and 127 of the 386 tree pairs with 30-39 leaves. We were able to reliably compute replug distances for trees with up to 65 leaves (114/133 successes in the 50-65 leaf range) and TBR distances for trees with up to 100 leaves (1 failure from the 261 tree pairs in the 50-99 leaf range).

8.3 Mean Distance
Finally, we compared the mean SPR, replug, and TBR distance values for trees with a given number of leaves (Fig. 7). We computed these values only for the tree pairs for which we successfully computed the SPR distance to obtain a fair comparison. We found that the replug distance is an excellent lower bound that closely tracks the SPR distance with a mean difference of 0.047 over 5335 tree pairs. The TBR distance had a mean difference of 0.325 from the SPR distance and 0.278 from the replug distance. In each of our completed tests, the TBR distance was at most 2 less than the replug distance and at most 3 less than the SPR distance, while the SPR distance never exceeded the replug distance by more than 1.
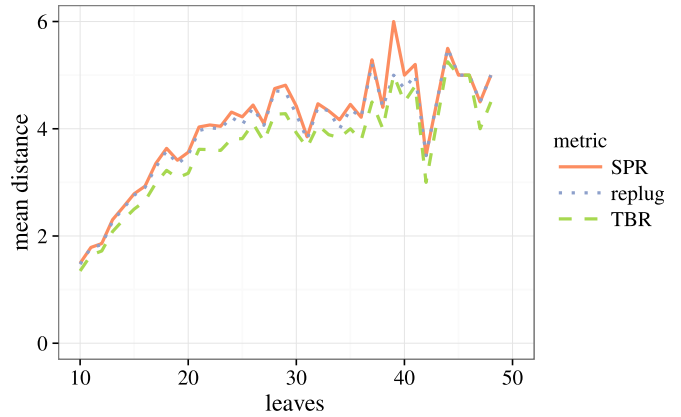
9 CONCLUSIONS
We have worked to extend understanding of and methods to calculate the SPR distance between unrooted trees in several directions. We identified four properties of optimal SPR paths between unrooted trees which are atypical of NP-hard tree distance metrics that can be efficiently solved with an MAF formulation: both endpoints of an edge may be moved, the same endpoint of an edge may be moved multiple times, common clusters are not necessarily maintained, and common paths may be broken. These observations suggest that an MAF formulation is not applicable for computing the SPR distance between unrooted trees.

To obtain an efficient search strategy we instead introduced a new lower bound on the SPR distance that we call the replug distance. Although the computational complexity of this distance is unknown, our work shows that it is fixed parameter tractable and we conjecture that it is NP-hard to compute. The replug distance captures important properties of the SPR distance while relaxing the requirement that intermediate structures be connected trees. Moreover, we showed that the replug distance can be modeled using an MAF variant that we call a maximum endpoint agreement forest or MEAF. We developed a two-phase fixed-parameter bounded search tree algorithm for the replug distance that runs in time $O(16^k(k^{1.5} + n))$-time, where k is the SPR distance between the trees and n their number of leaves. This algorithm works by exploring the set of maximal agreement forests of the trees. Experiments suggest that these sets are typically small in practice [56]. Each such forest is then refined to a maximal endpoint agreement forest by solving boolean monotone $\text{CNF}_+(\leq 2)$ sets of constraints. These formulas naturally arise by considering sets of structures we call dead components in the trees given an mAF, and can be solved in polynomial time when converted to the minimimum edge cover problem of an appropriate constraint graph.

Finally, we developed a new incremental heuristic search algorithm that we call progressive A* search. This algorithm expands the search for a given tree outward from the initial tree by applying increasingly expensive but more accurate lower bound estimators. The algorithm is applicable to any search problem that admits such a set of estimators, each of which is a lower bound on the next. Progressive A* search computes the SPR distance d between two unrooted trees in

$O((28d)!!16^d d^{1.5})$-time. Our implementation in the uspr software package uses a TBR approximation, TBR distance, and replug distance as lower bounds. Our results show that uspr is nearly two orders of magnitude faster than the previous best software for computing SPR distances between unrooted trees. In particular, our methods double the maximum SPR distance that can be computed given 5 hours from 7 to 14, and increase the size of trees that can be reliably compared from 30 to 50 leaves. Moreover, our implementation of the replug and TBR distance metrics can handle distances as large as 21 and 25, respectively. The replug and TBR algorithms were able to reliably handle trees with up to 65 and 100 leaves, respectively.

The development of initial fixed-parameter bounded search tree algorithms for the SPR distance between rooted trees quickly led to the current state of the art algorithms which can handle distances of 100 or more on trees with hundreds of leaves in only fractions of a second. These improvements came from a combination of structural insights leading to improved branching rules and new reductions such as the cluster reduction rule which splits the compared trees into independently comparable subtrees. Although unrooted trees are not clusterable with respect to the unrooted SPR distance, we conjecture that they are clusterable with respect to the replug distance. As faster algorithms for the TBR and replug distance will immediately reduce the time required by our progressive A* search framework, we believe that refining these algorithms represents the best strategy for futher reducing the time required to compute SPR distances between unrooted trees. In addition, techniques from the incremental heuristic search literature may also lead to improved algorithms for computing SPR distances. Alternatively, further structural analysis may lead to a direct refinement procedure from MEAFs to unrooted SPR paths, representing another branch of study for future work. Finally, it remains to extend our methods to nonbinary trees or to comparing sets of more than two trees, which are active avenues of research with respect to rooted trees.

ACKNOWLEDGMENTS

REFERENCES

[1] D. M. Hillis, C. Moritz, and B. K. Mable, Eds., *Molecular Systematics*. Sunderland, MA, USA: Sinauer Associates, 1996.

[2] E. V. Koonin, "The turbulent network dynamics of microbial evolution and the statistical tree of life," *J. Mol. Evol.*, vol. 80, no. 5/6, pp. 244–250, 2015.

[3] E. Castro-Nallar, M. Pérez-Losada, G. F. Burton, and K. A. Crandall, "The evolution of HIV: Inferences using phylogenetics," *Mol. Phylogenetics Evol.*, vol. 62, no. 2, pp. 777–792, 2012.

[4] B. F. Haynes, G. Kelsoe, S. C. Harrison, and T. B. Kepler, "B-cell-lineage immunogen design in vaccine development with HIV-1 as a case study," *Nature Biotechnology*, vol. 30, no. 5, pp. 423–433, 2012.

[5] M. R. Helmus, T. J. Bland, C. K. Williams, and A. R. Ives, "Phylogenetic measures of biodiversity," *Am. Naturalist*, vol. 169, no. 3, pp. E68–E83, 2007.

[6] F. A. Matsen, "Phylogenetics and the human microbiome," *Systematic Biol.*, vol. 64, no. 1, pp. e26–e41, 2015.

[7] N. Galtier and V. Daubin, "Dealing with incongruence in phylogenomic analyses," *Philosophical Trans. R. Soc. B: Biol. Sci.*, vol. 363, no. 1512, pp. 4023–4029, 2008.

[8] R. G. Beiko, T. J. Harlow, and M. A. Ragan, "Highways of gene sharing in prokaryotes," *Proc. Nat. Acad. Sci. USA*, vol. 102, no. 40, pp. 14 332–14 337, 2005.

[9] C. Whidden, N. Zeh, and R. G. Beiko, "Supertrees based on the subtree prune-and-regraft distance," *Systematic Biol.*, vol. 63, no. 4, pp. 566–581, 2014.

[10] D. Pisani, J. A. Cotton, and J. O. McInerney, "Supertrees disentangle the chimerical origin of eukaryotic genomes," *Mol. Biol. Evol.*, vol. 24, no. 8, pp. 1752–1760, 2007.

[11] M. Steel and A. Rodrigo, "Maximum likelihood supertrees," *Systematic Biol.*, vol. 57, no. 2, pp. 243–250, Apr. 2008. [Online]. Available: http://dx.doi.org/10.1080/10635150802033014

[12] M. S. Bansal, J. G. Burleigh, O. Eulenstein, and D. Fernández-Baca, "Robinson-Foulds supertrees," *Algorithms Mol. Biol.*, vol. 5, no. 18, pp. 1–12, 2010.

[13] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Math. Biosci.*, vol. 53, no. 1/2, pp. 131–147, 1981.

[14] W. H. E. Day, "Optimal algorithms for comparing trees with labeled leaves," *J. Classification*, vol. 2, no. 1, pp. 7–28, 1985.

[15] G. S. Brodal, R. Fagerberg, and C. N. Pedersen, "Computing the quartet distance between evolutionary trees in time O (n log n)," *Algorithmica*, vol. 38, no. 2, pp. 377–395, 2004.

[16] M. Owen and J. S. Provan, "A fast algorithm for computing geodesic distances in tree space," *IEEE/ACM Trans. Comput. Biol Bioinf.*, vol. 8, no. 1, pp. 2–13, Jan./Feb. 2011.

[17] M. Baroni, S. Grünewald, V. Moulton, and C. Semple, "Bounding the number of hybridisation events for a consistent evolutionary history," *J. Math. Biol.*, vol. 51, no. 2, pp. 171–182, 2005.

[18] T. C. Bruen and D. Bryant, "Parsimony via consensus," *Systematic Biol.*, vol. 57, no. 2, pp. 251–256, Apr. 1, 2008. [Online]. Available: http://sysbio.oxfordjournals.org/content/57/2/251.abstract

[19] S. Kelk and M. Fischer, "On the complexity of computing MP distance between binary phylogenetic trees," *Ann. Combinatorics*, vol. 21, no. 4, pp. 573–604, 2017.

[20] V. Moulton and T. Wu, "A parsimony-based metric for phylogenetic trees," *Adv. Appl. Math.*, vol. 66, pp. 22–45, 2015.

[21] R. G. Beiko and N. Hamilton, "Phylogenetic identification of lateral genetic transfer events," *BMC Evol. Biol.*, vol. 6, no. 1, 2006, Art. no. 15.

[22] W. P. Maddison, "Gene trees in species trees," *Systematic Biol.*, vol. 46, no. 3, pp. 523–536, 1997.

[23] L. Nakhleh, T. Warnow, C. R. Lindner, and K. St. John, "Reconstructing reticulate evolution in species—Theory and practice," *J. Comput. Biol.*, vol. 12, no. 6, pp. 796–811, 2005.

[24] M. N. Price, P. S. Dehal, and A. P. Arkin, "FastTree 2–Approximately maximum-likelihood trees for large alignments," *PLoS One*, vol. 5, no. 3, 2010, Art. no. e9490.

[25] A. Stamatakis, "RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models," *Bioinf.*, vol. 22, no. 21, pp. 2688–2690, 23 Aug. 2006.

[26] F. Ronquist, M. Teslenko, P. van der Mark, D. L. Ayres, A. Darling, S. Höhna, B. Larget, L. Liu, M. A. Suchard, and J. P. Huelsenbeck, "MrBayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space," *Systematic Biol.*, vol. 61, no. 3, pp. 539–542, 22 Feb. 2012. [Online]. Available: http://dx.doi.org/10.1093/sysbio/sys029

[27] R. Bouckaert, J. Heled, D. Kühnert, T. Vaughan, C.-H. Wu, D. Xie, M. A. Suchard, A. Rambaut, and A. J. Drummond, "BEAST 2: A software platform for Bayesian evolutionary analysis," *PLoS Comput. Biol.*, vol. 10, no. 4, 2014, Art. e1003537.

[28] R. Atkins and C. McDiarmid, "Extremal distances for subtree transfer operations in binary trees," arXiv:1509.00669, 2015. [Online]. Available: http://arxiv.org/abs/1509.00669

[29] Y. Ding, S. Grünewald, and P. J. Humphries, "On agreement forests," *J. Combinatorial Theory, Series A*, vol. 118, no. 7, pp. 2059–2065, Oct. 2011.

[30] C. Lakner, P. Van Der Mark, J. P. Huelsenbeck, B. Larget, and F. Ronquist, "Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics," *Systematic Biol.*, vol. 57, no. 1, pp. 86–103, 2008.

[31] S. Höhna and A. J. Drummond, "Guided tree topology proposals for Bayesian phylogenetic inference," *Systematic Biol.*, vol. 61, no. 1, pp. 1–11, 2012.

[32] C. Whidden and F. A. Matsen IV , "Quantifying MCMC exploration of phylogenetic tree space," *Systematic Biol.*, vol. 64, no. 3, pp. 472–491, 27 Jan. 2015. [Online]. Available: http://dx.doi.org/10.1093/sysbio/syv006

[33] M. Bordewich and C. Semple, "Computing the minimum number of hybridization events for a consistent evolutionary history," *Discrete Appl. Math.*, vol. 155, no. 8, pp. 914–928, 2007.

[34] M. Bordewich and C. Semple, "On the computational complexity of the rooted subtree prune and regraft distance," *Ann. Combinatorics*, vol. 8, no. 4, pp. 409–423, 2005.

[35] G. Hickey, F. Dehne, A. Rau-Chaplin, and C. Blouin, "SPR distance computation for unrooted trees," *Evol. Bioinf. Online*, vol. 4, 2008, Art. no. 17.

[36] C. Whidden, R. G. Beiko, and N. Zeh, "Fast FPT algorithms for computing rooted agreement forests: Theory and experiments," in *Proc. Int. Symp. Exp. Algorithms*, 2010, pp. 141–153.

[37] G. Dudas, T. Bedford, S. Lycett, and A. Rambaut, "Reassortment between Influenza B lineages and the emergence of a co-adapted PB1-PB2-HA gene complex," *Mol. Biol. Evol.*, vol. 32, pp. 162–172, 2014.

[38] C. Whidden and F. A. Matsen IV, "Ricci-Ollivier curvature of the rooted phylogenetic subtree-prune-regraft graph," *Theoretical Comput. Sci.*, vol. 699, pp. 1–20, 2017.

[39] J. Hein, T. Jiang, L. Wang, and K. Zhang, "On the complexity of comparing evolutionary trees," *Discrete Appl. Math.*, vol. 71, no. 1–3, pp. 153–169, 1996.

[40] B. L. Allen and M. Steel, "Subtree transfer operations and their induced metrics on evolutionary trees," *Ann. Combinatorics*, vol. 5, no. 1, pp. 1–15, 2001.

[41] Y. Wu, "A practical method for exact computation of subtree prune and regraft distance," *Bioinf.*, vol. 25, no. 2, pp. 190–196, 2009.

[42] M. L. Bonet and K. St . John, "Efficiently calculating evolutionary tree measures using SAT," in *Proc. 12th Int. Conf. Theory Appl. Satisfiability Testing*, 2009, pp. 4–17.

[43] C. Whidden, R. G. Beiko, and N. Zeh, "Fixed-Parameter algorithms for maximum agreement forests," *SIAM J. Comput.*, vol. 42, no. 4, pp. 1431–1466, 2013.

[44] F. Shi, J. You, and Q. Feng, "Improved approximation algorithm for maximum agreement forest of two trees," in *Proc. Frontiers Algorithmics*, 2014, pp. 205–215.

[45] Z.-Z. Chen, Y. Fan, and L. Wang, "Faster exact computation of rSPR distance," *J. Combinatorial Optim.*, vol. 29, no. 3, pp. 605–635, 2013.

[46] C. Whidden, R. G. Beiko, and N. Zeh, "Fixed-Parameter and approximation algorithms for maximum agreement forests of multifurcating trees," *Algorithmica*, vol. 74, pp. 1–36, 2015, doi: 10.1007/s00453-015-9983-z.

[47] J. Chen, J.-H. Fan, and S.-H. Sze, "Parameterized and approximation algorithms for maximum agreement forest in multifurcating trees," *Theoretical Comput. Sci.*, vol. 562, pp. 496–512, 2015.

[48] F. Shi, J. Chen, Q. Feng, and J. Wang, "Approximation algorithms for maximum agreement forest on multiple trees," in *Proc. Comput. Combinatorics: 20th Int. Conf.*, 2014, vol. 8591, 2014, Art. no. 381.

[49] S. Linz and C. Semple, "A cluster reduction for computing the subtree distance between phylogenies," *Ann. Combinatorics*, vol. 15, no. 3, pp. 465–484, 2011.

[50] C. Whidden and F. A. Matsen IV, "Chain reduction preserves the unrooted subtree prune-and-regraft distance," arXiv:1611.02351, 2016. [Online]. Available: http://arxiv.org/abs/1611.02351

[51] M. L. Bonet and K. St John, "On the complexity of uSPR distance," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 7, no. 3, pp. 572–576, Jul.-Sep. 2010.

[52] F. Shi, J. Chen, Q. Feng, and J. Wang, "Paramaterized algorithms for maximum agreement forest on multiple trees," in *Proc. Comput. Combinatorics: 19th Int. Conf.*, 2013, vol. 8591, pp. 567–578.

[53] M. Baroni, C. Semple, and M. Steel, "Hybrids in real time," *Systematic Biol.*, vol. 55, no. 1, pp. 46–56, 2006.

[54] C. Whidden and N. Zeh, "A unifying view on approximation and FPT of agreement forests," in *Proc. 9th Int. Workshop*, 2009, vol. 5724, 2009, pp. 390–401.

[55] J. Chen, J.-H. Fan, and S.-H. Sze, "Parameterized and approximation algorithms for the MAF problem in multifurcating trees," in *Proc. Graph-Theoretic Concepts Comput. Sci.*, 2013, pp. 152–164.

[56] J. Voorkamp, "Maximal acyclic agreement forests," *J. Comput. Biol.*, vol. 21, no. 10, pp. 723–731, 2014.

[57] S. Porschen and E. Speckenmeyer, "Algorithms for variable-weighted 2-SAT and dual problems," in *Proc. Theory Appl. Satisfiability Testing*, 2007, pp. 173–186.

[58] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[59] C. Whidden, "uspr," 2015. [Online]. Available: https://github.com/cwhidden/uspr.

[60] Y. Wu and J. Wang, "Fast computation of the exact hybridization number of two phylogenetic trees," in *Proc. Bioinf. Res. Appl.*, 2010, pp. 203–214.

[61] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA, USA: Freeman, 1979.

[62] S. Micali and V. V. Vazirani, "An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs," in *Proc. 21st Annu. Symp. Found. Comput. Sci.*, 1980, pp. 17–27.

Chris Whidden received the PhD degree in computer science from Dalhouse University, Halifax, Canada, in 2013. He is currently a postdoctoral research fellow in the Fred Hutchinson Cancer Research Center, Seattle, WA. His research interests include developing efficient algorithms and software for solving NP-hard problems, with a particular focus on inferring and comparing phylogenetic trees and networks.

Frederick A. Matsen IV received the PhD degree in mathematics from Harvard University, Cambridge, MA, in 2006. He was a postdoctoral scholar with the University of Canterbury, New Zealand, from 2006 to 2007 and UC Berkeley, Berkeley, California, from 2007 to 2010. He is an associate member in the Fred Hutchinson Cancer Research Center, Seattle, WA. His research interests include developing new methods to analyze metagenomic, viral, and immune cell sequence data, as well as pursuing more abstract methodological questions in evolutionary tree reconstruction.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.