Constructing a Consensus Phylogeny from a Leaf-Removal Distance (Extended Abstract)

Chauve, Cedric; Jones, Mark; Lafond, Manuel; Scornavacca, Celine; Weller, Mathias

Constructing a Consensus Phylogeny from a Leaf-Removal Distance (Extended Abstract)*

Cedric Chauve[1], Mark Jones[2], Manuel Lafond[3], Céline Scornavacca[4], and Mathias Weller[5]

[1] Department of Mathematics
Simon Fraser University
Burnaby, Canada
cedric.chauve@sfu.ca
[2] Delft Institute of Applied Mathematics
Delft University of Technology
P.O. Box 5, 2600 AA, Delft, the Netherlands
M.E.L.Jones@tudelft.nl
[3] Department of Mathematics and Statistics
University of Ottawa
Ottawa, Canada
mlafond2@uOttawa.ca
[4] Institut des Sciences de l'Evolution
Université de Montpellier, CNRS, IRD, EPHE
Montpellier - France
Celine.Scornavacca@umontpellier.fr
[5] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université de Montpellier, IBC
Montpellier - France
mathias.weller@lirmm.fr

Abstract. Understanding the evolution of a set of genes or species is a fundamental problem in evolutionary biology. The problem we study here takes as input a set of trees describing possibly discordant evolutionary scenarios for a given set of genes or species, and aims at finding a single tree that minimizes the leaf-removal distance to the input trees. This problem is a specific instance of the general consensus/supertree problem, widely used to combine or summarize discordant evolutionary trees. The problem we introduce is specifically tailored to address the case of discrepancies between the input trees due to the misplacement of individual taxa. Most supertree or consensus tree problems are computationally intractable, and we show that the problem we introduce is also NP-hard. We provide tractability results in form of a 2-approximation algorithm and a parameterized algorithm with respect to the number of removed leaves. We also introduce a variant that minimizes the maximum number d of leaves that are removed from any input tree, and provide a parameterized algorithm for this problem with parameter d.

* All missing proofs are provided in [6].

Keywords: Computational biology · Phylogenetics · Parameterized algorithms · Approximation · Consensus trees · Leaf deletion.

1 Introduction

In the present paper, we consider a very generic computational biology problem: given a collection of trees representing, possibly discordant, evolutionary scenarios for a set of biological entities (genes or species – also called *taxa* in the following), we want to compute a single tree that agrees as much as possible with the input trees. Several questions in computational biology can be phrased in this generic framework. For example, for a given set of homologous gene sequences that have been aligned, one can sample *evolutionary trees* for this gene family according to a well defined posterior distribution and then ask how this collection of trees can be combined into a single gene tree, a problem known as *tree amalgamation* [16]. In phylogenomics, one aims at *inferring a species tree* from a collection of input trees obtained from whole-genome sequence data. A first approach considers gene families and proceeds by computing individual *gene trees* from a large set of gene families, and then combining this collection of gene trees into a unique species tree for the given set of taxa; this requires handling the discordant signal observed in the gene trees due to evolutionary processes such as gene duplication and loss [13], lateral gene transfer [17], or incomplete lineage sorting [15]. Another approach concatenates the sequence data into a single large multiple sequence alignment, that is then partitioned into overlapping subsets of taxa for which partial evolutionary trees are computed, and a unique species tree is then inferred by combining the resulting collection of partial trees [14].

For example, the Maximum Agreement Subtree (MAST) problem considers a collection of input trees[1], all having the same leaf labels and looks for a tree of maximum size (number of leaves), which agrees with each of the input trees. This problem is tractable for trees with bounded degree but NP-hard generally [2]. The MAST problem is a *consensus problem*, because the input trees share the same leaf labels set, and the output tree is called a *consensus* tree. In the *supertree framework*, the input trees might not all have identical label sets, but the output is a tree on the whole label set, called a *supertree*. For example, in the Robinson-Foulds (RF) supertree problem, the goal is to find a supertree that minimizes the sum of the RF-distances to the individual input trees [18]. One way to compute consensus trees and supertrees that is closely related to our work is to modify the collection of input trees minimally in such a way that the resulting modified trees all agree. For example, in the MAST problem, modifications of the input trees consist in removing a minimum number of taxa from the whole label set, while in the Agreement Supertree by Edge Contraction (AST-EC) problem, one is asked to contract a minimum number of edges of the input trees such that the resulting (possibly non-binary) trees

[1] All trees we consider here are uniquely leaf-labeled, rooted (*i.e.* are out-trees) and binary; see next section for formal definitions.

all agree with at least one supertree [10]; in the case where the input trees are all triplets (rooted trees on three leaves), this supertree problem is known as the Minimum Rooted Triplets Inconsistency problem [5]. The SPR Supertree problem considers a similar problem where the input trees can be modified with the Subtree-Prune-and-Regraft (SPR) operator [19].

In the present work, we introduce a new consensus problem, called `LR-Consensus`. Given a collection of input trees having the same leaf labels set, we want to remove a minimum number of leaves – an operation called a Leaf-Removal (LR) – from the input trees such that the resulting pruned trees all agree. Alternatively, this can be stated as finding a consensus tree that minimizes the cumulated *leaf-removal distance* to the collection of input trees. This problem also applies to tree amalgamation and to species tree inference from one-to-one orthologous gene families, where the LR operation aims at correcting the misplacement of a single taxon in an input tree. This may occur particularly in the case of 'rogue taxa' [1], for example when a sequence from a taxon has mistakenly been put in a gene family where it does not belong.

In the next section, we formally define the problems we consider, and how they relate to other supertree problems. Next we show that the `LR-Consensus` problem is NP-hard and that in some instances, a large number of leaves need to be removed to lead to a consensus tree. We then provide a 2-approximation algorithm, and show that the problem is fixed-parameter tractable (FPT) when parameterized by the total number of LR. However, these FPT algorithms have impractical time complexity, and thus, to answer the need for practical algorithms, we introduce a variant of the `LR-Consensus` problem, where we ask if a consensus tree can be obtained by removing at most d leaves from each input tree, and describe an FPT algorithm with parameter d.

2 Preliminary Notions and Problem Statements

Trees. All trees in the rest of the document are assumed to be rooted and binary. If T is a tree, we denote its root by $r(T)$ and its leaf set by $\mathcal{L}(T)$. Each leaf is labeled by a distinct element from a *label set* $\mathcal{X}$, and we denote by $\mathcal{X}(T)$ the set of labels of the leaves of T. We may sometimes use $\mathcal{L}(T)$ and $\mathcal{X}(T)$ interchangeably. For some $X \subseteq \mathcal{X}$, we denote by $lca_T(X)$ the *least common ancestor* of X in T. The subtree rooted at a node $u \in V(T)$ is denoted T_u and we may write $\mathcal{L}_T(u)$ for $\mathcal{L}(T_u)$. If T_1 and T_2 are two trees and e is an edge of T_1, grafting T_2 on e consists of subdividing e and letting the resulting degree 2 node become the parent of $r(T_2)$. Grafting T_2 above T_1 consists of creating a new node r, then letting r become the parent of $r(T_1)$ and $r(T_2)$ (note that grafting T_2 above T_1 is equivalent to grafting T_1 above T_2). Grafting T_2 on T_1 means grafting T_2 either on an edge of T_1 or above T_1.

The Leaf Removal Operation. For a subset $L \subseteq \mathcal{X}$, we denote by $T - L$ the tree obtained from T by removing every leaf labeled by L, contracting the resulting non-root vertices of degree two, and repeatedly deleting the resulting root vertex

while it has degree one. The *restriction* $T|_L$ of T to L is the tree $T-(\mathcal{X}\setminus L)$, *i.e.* the tree obtained by removing every leaf *not* in L. A *triplet* is a rooted tree on 3 leaves. We denote a triplet R with leaf set $\{a,b,c\}$ by $ab|c$ if c is the leaf that is a direct child of the root (the parent of a and b being its other child). We say $R=ab|c$ is a triplet of a tree T if $T|_{\{a,b,c\}}=R$. We denote $tr(T)=\{ab|c : ab|c$ is a triplet of $T\}$.

We define a *distance function* d_{LR} between two trees T_1 and T_2 on the same label set $\mathcal{X}$ consisting in the minimum number of labels to remove from $\mathcal{X}$ so that the two trees are equal. That is,

$$d_{LR}(T_1,T_2)=\min\{|X| : X\subseteq\mathcal{X} \text{ and } T_1-X=T_2-X\}$$

Note that d_{LR} is closely related to the Maximum Agreement Subtree (MAST) between two trees on the same label set $\mathcal{X}$, which consists in a subset $X'\subseteq\mathcal{X}$ of maximum size such that $T_1|_{X'}=T_2|_{X'}$: $d_{LR}(T_1,T_2)=|\mathcal{X}|-|X'|$. The MAST of two binary trees on the same label set can be computed in time $O(n\log n)$, where $n=|\mathcal{X}|$ [8], and so d_{LR} can be found within the same time complexity.

Problem Statements. In this paper, we are interested in finding a tree T on $\mathcal{X}$ minimizing the sum of d_{LR} distances to a given set of input trees.

`LR-Consensus`
Given: a set of trees $\mathcal{T}=\{T_1,\ldots,T_t\}$ with $\mathcal{X}(T_1)=\ldots=\mathcal{X}(T_t)=\mathcal{X}$.
Find: a tree T on label set $\mathcal{X}$ that minimizes $\sum_{T_i\in\mathcal{T}} d_{LR}(T,T_i)$.

We can reformulate the `LR-Consensus` problem as the problem of removing a minimum number of leaves from the input trees so that they are *compatible*. Although the equivalence between both formulations is obvious, the later formulation will often be more convenient. We need to introduce more definitions in order to establish this equivalence.

A set of trees $\mathcal{T}=\{T_1,\ldots,T_t\}$ is called *compatible* if there is a tree T such that $\mathcal{X}(T)=\bigcup_{T_i\in\mathcal{T}}\mathcal{X}(T_i)$ and $T|_{\mathcal{X}(T_i)}=T_i$ for every $i\in[t]$. In this case, we say that T *displays* $\mathcal{T}$. A list $\mathcal{C}=(\mathcal{X}_1,\ldots,\mathcal{X}_t)$ of subsets of $\mathcal{X}$ is a *leaf-disagreement* for $\mathcal{T}$ if $\{T_1-\mathcal{X}_1,\ldots,T_t-\mathcal{X}_t\}$ is compatible. The *size* of $\mathcal{C}$ is $\sum_{i\in[t]}|\mathcal{X}_i|$. We denote by $AST_{LR}(\mathcal{T})$ the minimum size of a leaf-disagreement for $\mathcal{T}$, and may sometimes write $AST_{LR}(T_1,\ldots,T_t)$ instead of $AST_{LR}(\mathcal{T})$. A subset $\mathcal{X}'\subseteq\mathcal{X}$ of labels is a *label-disagreement* for $\mathcal{T}$ if $\{T_1-\mathcal{X}',\ldots,T_t-\mathcal{X}'\}$ is compatible. Note that if $\mathcal{T}=\{T_1,T_2\}$, then the minimum size of a leaf-disagreement and label-disagreement for $\mathcal{T}$ are the same, namely $d_{LR}(T_1,T_2)$. Note however that this does not hold in general (see Figure 1 for an example). We may now define the `AST-LR` problem.

`Agreement Subtrees by Leaf-Removals` (`AST-LR`)
Given: a set of trees $\mathcal{T}=\{T_1,\ldots,T_t\}$ with $\mathcal{X}(T_1)=\ldots=\mathcal{X}(T_t)=\mathcal{X}$.
Find: a leaf-disagreement $\mathcal{C}$ for $\mathcal{T}$ of minimum size.

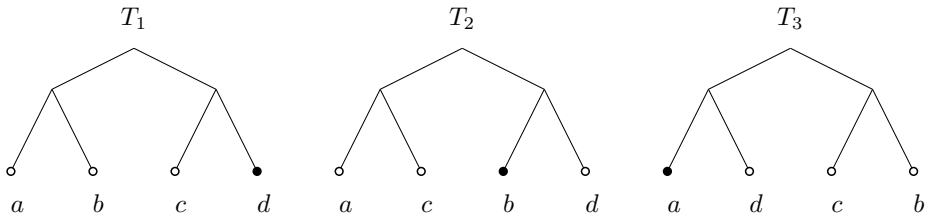

Fig. 1: Example instance $\mathcal{T} = \{T_1, T_2, T_3\}$ of `AST-LR` with label set $\mathcal{X} = \{a, b, c, d\}$. The list $(\mathcal{X}_1 = \{d\}, \mathcal{X}_2 = \{b\}, \mathcal{X}_3 = \{a\})$ is a leaf-disagreement for $\mathcal{T}$ of size 3. The set $\mathcal{X}' = \{a, b\}$ is a label-disagreement of size 2. Note that there is no leaf-disagreement for $\mathcal{T}$ of size 2.

Lemma 1. *Let $\mathcal{T} = \{T_1, \ldots, T_t\}$ be a set of trees on the same label set $\mathcal{X}$, with $n = |\mathcal{X}|$. Given a supertree T such that $v := \sum_{T_i \in \mathcal{T}} d_{LR}(T, T_i)$, one can compute in time $O(tn \log(n))$ a leaf-disagreement $\mathcal{C}$ of size at most v. Conversely, given a leaf-disagreement $\mathcal{C}$ for $\mathcal{T}$ of size v, one can compute in time $O(tn \log^2(tn))$ a supertree T such that $\sum_{T_i \in \mathcal{T}} d_{LR}(T, T_i) \leq v$.*

Proof. In the first direction, for each $T_i \in \mathcal{T}$, there is a set $X_i \subseteq \mathcal{X}$ of size $d_{LR}(T, T_i)$ such that $T_i - X_i = T - X_i$. Moreover, X_i can be found in time $O(n \log n)$. Thus $(X_1, \ldots, X_t)$ is a leaf-disagreement of the desired size and can be found in time $O(tn \log n)$. Conversely, let $\mathcal{C} = (X_1, \ldots, X_t)$ be a leaf-disagreement of size v. As $\mathcal{T}' = \{T_1 - X_1, \ldots, T_t - X_t\}$ is compatible, there is a tree T that displays $\mathcal{T}'$, and it is easy to see that the sum of distances between T and $\mathcal{T}'$ is at most the size of $\mathcal{C}$. As for the complexity, it is shown in [9] how to compute in time $O(tn \log^2(tn))$, given a set of trees $\mathcal{T}'$, a tree T displaying $\mathcal{T}'$ if one exists. □

From Lemma 1, both problems share the same optimality value, the NP-hardness of one implies the hardness of the other, and approximating one problem within a factor c implies that the other problem can be approximated within a factor c. We conclude this subsection with the introduction of a parameterized variant of the `AST-LR` problem.

`AST-LR-d`
Input: a set of trees $\mathcal{T} = \{T_1, \ldots, T_t\}$ with $\mathcal{X}(T_1) = \ldots = \mathcal{X}(T_t) = \mathcal{X}$, and an integer d.
Question: Are there $\mathcal{X}_1, \ldots, \mathcal{X}_t \subseteq \mathcal{X}$ such that $|\mathcal{X}_i| \leq d$ for each $i \in [t]$, and $\{T_1 - \mathcal{X}_1, \ldots, T_t - \mathcal{X}_t\}$ is compatible?

We call a tree T^* a *solution* to the `AST-LR-d` instance if $d_{LR}(T_i, T^*) \leq d$ for each $i \in [t]$.

Relation to Other Supertree/Consensus Tree Problems. The most widely studied supertree problem based on modifying the input trees is the SPR Supertree problem, where arbitrarily large subtrees can be moved in the input trees to

make them all agree (see [19] and references there). The interest of this problem is that the SPR operation is very general, modelling lateral gene transfer and introgression. The LR operation we introduce is a limited SPR, where the displaced subtree is composed of a single leaf. An alternative to the SPR operation to move subtrees within a tree is the Edge Contraction (EC) operation, that contracts an edge of an input tree, thus increasing the degree of the parent node. This operation allows correcting the local misplacement of a full subtree. `AST-EC` is NP-complete but can be solved in $O((2t)^p tn^2)$ time where p is the number of required EC operations [10].

Compared to the two problems described above, an LR models a very specific type of error in evolutionary trees, that is the misplacement of a single taxon (a single leaf) in one of the input trees. This error occurs frequently in reconstructing evolutionary trees, and can be caused for example by some evolutionary process specific to the corresponding input tree (recent incomplete lineage sorting, or recent lateral transfer for example). Conversely, it is not well adapted to model errors, due for example to ancient evolutionary events that impact large subtrees. However, an attractive feature of the LR operation is that computing the LR distance is equivalent to computing the `MAST` cost and is thus tractable, unlike the SPR distance which is hard to compute. This suggests that the `LR-Consensus` problem might be easier to solve than the SPR Supertree problem, and we provide indeed several tractability results. Compared to the `AST-EC` problem, the `AST-LR` problem is naturally more adapted to correct single taxa misplacements as the EC operation is very local and the number of EC required to correct a taxon misplacement is linear in the length of the path to its correct location, while the LR cost of correcting this is unitary. Last, `LR-Consensus` is more flexible than the `MAST` problem as it relies on modifications of the input trees, while with the way `MAST` corrects a misplaced leaf requires to remove this leaf from all input trees. This shows that the problems `AST-LR` and `AST-LR-d` complement well the existing corpus of gene tree correction models.

3 Hardness and Approximability of `AST-LR`

In this section, we show that the `AST-LR` problem is NP-hard, from which the `LR-Consensus` hardness follows. We then describe a simple factor 2 approximation algorithm. The algorithm turns out to be useful for analyzing the worst case scenario for `AST-LR` in terms of the required number of leaves to remove, as we show that there are `AST-LR` instances that require removing about $n - \sqrt{n}$ leaves in each input tree.

NP-Hardness of `AST-LR`

We assume here that we are considering the decision version of `AST-LR`, *i.e.* deciding whether there is a leaf-disagreement of size at most ℓ for a given ℓ. We use a reduction from the `MinRTI` problem: given a set $\mathcal{R}$ of rooted triplets, find a subset $\mathcal{R}' \subset \mathcal{R}$ of minimum cardinality such that $\mathcal{R} \setminus \mathcal{R}'$ is compatible. The `MinRTI`

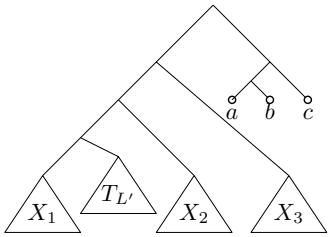

Fig. 2: Construction of the tree T_1 for an instance $\mathcal{R} = \{R_1, R_2, R_3\}$ of `MinRTI` in which $R_1 = ab|c$.

problem is NP-Hard (and furthermore $W[2]$-hard) [5], and hard to approximate within a $O(2^{\log^{1-\epsilon} n})$ factor [7]. Denote by $MINRTI(\mathcal{R})$ the minimum number of triplets to remove from $\mathcal{R}$ to attain compatibility. We describe the reduction here.

Let $\mathcal{R} = \{R_1, \ldots, R_t\}$ be an instance of `MinRTI`, with the label set $L := \bigcup_{i=1}^{t} \mathcal{X}(R_i)$. For a given integer m, we construct an `AST-LR` instance $\mathcal{T} = \{T_1, \ldots, T_t\}$ such that $MINRTI(\mathcal{R}) \leq m$ if and only if $AST_{LR}(\mathcal{T}) \leq t(|L| - 3) + m$.

We first construct a tree Z with additional labels which will serve as our main gadget. Let $\{L_i\}_{1 \leq i \leq t}$ be a collection of t new label sets, each of size $(|L|t)^{10}$, all disjoint from each other and all disjoint from L. Each tree in our `AST-LR` instance will be on label set $\mathcal{X} = L \cup L_1 \cup \ldots \cup L_t$. For each $i \in [t]$, let X_i be any tree with label set L_i. Obtain Z by taking any tree on t leaves $l_1, \ldots, l_t$, then replacing each leaf l_i by the X_i tree (*i.e.* l_i is replaced by $r(X_i)$). Denote by $r_Z(X_i)$ the root of the X_i subtree in Z.

Then for each $i \in [t]$, we construct T_i from R_i as follows. Let $L' = L \setminus \mathcal{X}(R_i)$ be the set of labels not appearing in R_i, noting that $|L'| = |L| - 3$. Let $T_{L'}$ be any tree with label set L', and obtain the tree Z_i by grafting $T_{L'}$ on the edge between $r_Z(X_i)$ and its parent. Finally, T_i is obtained by grafting R_i above Z_i. See Figure 2 for an example. Note that each tree T_i has label set $\mathcal{X}$ as desired. Also, it is not difficult to see that this reduction can be carried out in polynomial time. This construction can now be used to show the following.

Theorem 2. *The `AST-LR` and `LR-Consensus` problems are NP-hard.*

The idea of the proof is to show that in the constructed `AST-LR` instance, we are "forced" to solve the corresponding `MinRTI` instance. In more detail, we show that $MINRTI(\mathcal{R}) \leq m$ if and only if $AST_{LR}(\mathcal{T}) \leq t(|L| - 3) + m$. In one direction, given a set $\mathcal{R}'$ of size m such that $\mathcal{R} \setminus \mathcal{R}'$ is compatible, one can show that the following leaf removals from $\mathcal{T}$ make it compatible: remove, from each T_i, the leaves $L' = L \setminus \mathcal{X}(R_i)$ that were inserted into the Z subtree, then for each $R_i \in \mathcal{R}'$, remove a single leaf in $\mathcal{X}(R_i)$ from T_i. This sums up to $t(|L| - 3) + m$ leaf removals. Conversely, it can be shown that there always exists an optimal solution for $\mathcal{T}$ that removes, for each T_i, all the leaves $L' = L \setminus \mathcal{X}(R_i)$ inserted

in the Z subtree, plus an additional single leaf l from m trees $T_{i_1}, \ldots, T_{i_m}$ such that $l \in L$. The corresponding triplets $R_{i_1}, \ldots, R_{i_m}$ can be removed from $\mathcal{R}$ so that it becomes compatible.

Approximating AST-LR and Bounding Worst-Case Scenarios

Given the above result, it is natural to turn to approximation algorithms in order to solve AST-LR or LR-Consensus instances. It turns out that there is a simple factor 2 approximation for LR-Consensus which is achieved by interpreting the problem as finding a median in a metric space. Indeed, it is not hard to see that d_{LR} is a metric (over the space of trees on the same label set $\mathcal{X}$). A direct consequence, using an argument akin to the one in [12, p.351], is the following.

Theorem 3. *The following is a factor* 2 *approximation algorithm for* LR-Consensus*: return the tree* $T \in \mathcal{T}$ *that minimizes* $\sum_{T_i \in \mathcal{T}} d_{LR}(T, T_i)$*.*

Proof. Let T^* be an optimal solution for LR-Consensus, *i.e.* T^* is a tree minimizing $\sum_{T_i \in \mathcal{T}} d_{LR}(T_i, T^*)$, and let T be chosen as described in the theorem statement. Moreover let T' be the tree of $\mathcal{T}$ minimizing $d_{LR}(T', T^*)$. By the triangle inequality,

$$\sum_{T_i \in \mathcal{T}} d_{LR}(T', T_i) \leq \sum_{T_i \in \mathcal{T}} (d_{LR}(T', T^*) + d_{LR}(T^*, T_i)) \leq 2 \sum_{T_i \in \mathcal{T}} d_{LR}(T^*, T_i)$$

where the last inequality is due to the fact that $d_{LR}(T', T^*) \leq d_{LR}(T^*, T_i)$ for all i, by our choice of T'. Our choice of T implies $\sum_{T_i \in \mathcal{T}} d_{LR}(T, T_i) \leq \sum_{T_i \in \mathcal{T}} d_{LR}(T', T_i) \leq 2 \sum_{T_i \in \mathcal{T}} d_{LR}(T_i, T^*)$. □

Theorem 3 can be used to lower-bound the 'worst' possible instance of AST-LR. We show that in some cases, we can only keep about $\sqrt{|\mathcal{X}|}$ leaves per tree. That is, there are instances for which $AST_{LR}(\mathcal{T}) = \Omega(t(n - \sqrt{n}))$, where t is the number of trees and $n = |\mathcal{X}|$. The argument is based on a probabilistic argument, for which we will make use of the following result [4, Theorem 4.3.iv].

Theorem 4 ([4]). *For any constant* $c > e/\sqrt{2}$*, there is some* n_0 *such that for all* $n \geq n_0$*, the following holds: if* T_1 *and* T_2 *are two binary trees on* n *leaves chosen randomly, uniformly and independently, then* $\mathbb{E}[d_{LR}(T_1, T_2)] \geq n - c\sqrt{n}$*.*

Corollary 5. *There are instances of* AST-LR *in which* $\Omega(t(n - \sqrt{n}))$ *leaves need to be deleted.*

The above is shown by demonstrating that, by picking a set $\mathcal{T}$ of t random trees, the expected optimal sum of distances $\min_T \sum_{T_i \in \mathcal{T}} d_{LR}(T, T_i)$ is $\Omega(t(n - \sqrt{n})$. This is not direct though, since the tree T^* that minimizes this sum is not itself random, and so we cannot apply Theorem 4 directly on T^*. We can however, show that the tree $T' \in \mathcal{T}$ obtained using the 2-approximation, which is random, has expected sum of distances $\Omega(t(n - \sqrt{n}))$. Since T^* requires, at best, half the leaf deletions of T', the result follows. Note that finding a non-trivial upper bound on $AST_{LR}(\mathcal{T})$ is open.

4 Fixed-Parameter Tractability of AST-LR and AST-LR-d.

An alternative way to deal with computational hardness is parameterized complexity. In this section, we first show that AST-LR is fixed-parameter-tractable with respect to $q := AST_{LR}(\mathcal{T})$. More precisely, we show that AST-LR can be solved in $O(12^q tn^3)$ time, where $n := |\mathcal{X}|$. We then consider an alternative parameter d, and show that finding a tree T^*, if it exists, such that $d_{LR}(T_i, T^*) \leq d$ for every input tree T_i, can be done in $O(c^d d^{3d}(n^3 + tn \log n))$ time for some constant c.

4.1 Parameterization by q

The principle of the algorithm is the following. It is known that a set of trees $\mathcal{T} = \{T_1, \ldots, T_t\}$ is compatible if and only if the union of their triplet decomposition $tr(\mathcal{T}) = \bigcup_{T_i \in \mathcal{T}} tr(T_i)$ is compatible [3]. In a step-by-step fashion, we identify a conflicting set of triplets in $tr(\mathcal{T})$, each time branching into the (bounded) possible leaf-removals that can resolve the conflict. We stop when either $tr(\mathcal{T})$ is compatible after the performed leaf-removals, or when more than q leaves were deleted.

We employ a two phase strategy. In the first phase, we eliminate direct conflicts in $tr(\mathcal{T})$, *i.e.* if at least two of $ab|c, ac|b$ and $bc|a$ appear in $tr(\mathcal{T})$, then we recursively branch into the three ways of choosing one of the 3 triplets, and remove one leaf in each T_i disagreeing with the chosen triplet (we branch into the three possible choices, either removing a, b or c). The chosen triplet is locked in $tr(\mathcal{T})$ and cannot be changed later.

When the first phase is completed, there are no direct conflicts and $tr(\mathcal{T})$ consists of a *full set of triplets* on $\mathcal{X}$. That is, for each distinct $a, b, c \in \mathcal{X}$, $tr(\mathcal{T})$ contains exactly one triplet on label set $\{a, b, c\}$. Now, a full set of triplets is not necessarily compatible, and so in the second phase we modify $tr(\mathcal{T})$, again deleting leaves, in order to make it compatible. Only the triplets that have not been locked previously can be modified. This second phase is analogous to the FPT algorithm for *dense* MinRTI presented in [11]. The dense MinRTI is a variant of the MinRTI problem, introduced in Section 3, in which the input is a full set of triplets and one has to decide whether p triplets can be deleted to attain compatibility.

Theorem 6 ([11]). *A full set of triplets $\mathcal{R}$ is compatible if and only if for any set of four labels $\{a, b, c, d\}$, $\mathcal{R}$ does not contain the subset $\{ab|c, cd|b, bd|a\}$ nor the subset $\{ab|c, cd|b, ad|b\}$.*

One can check, through an exhaustive enumeration of the possibilities, that given a conflicting set of triplets R_1, R_2, R_3 where $R_1 = ab|c, R_2 = cd|b, R_3 \in \{bd|a, ad|b\}$, any tree on a set $\mathcal{X}$ containing $\{a, b, c, d\}$ must have at least one of the following triplets: (1) $bc|a$; (2)$ac|b$; (3)$bd|c$; (4) $ab|d$. Note that each of these conflicts with one of R_1, R_2, R_3. This leads to a $O(4^p n^3)$ algorithm for solving dense MinRTI: find a conflicting set of four labels, and branch on the four possibilities, locking the selected triplet each time.

For the second phase of AST-LR, we propose a slight variation of this algorithm. Each time a triplet R is chosen and locked, say $R = ab|c$, the trees containing $ac|b$ or $bc|a$ must loose a, b or c. We branch into these three possibilities. Thus for each conflicting 4-set, there are four ways of choosing a triplet, then for each such choice, three possible leaves to delete from a tree. This gives 12 choices to branch into recursively. Algorithm 1 summarises the procedure and its analysis yields the following.

Theorem 7. *AST-LR can be solved in time* $O(12^q tn^3)$.

Data: $\mathcal{T}$ is the set of input trees, q is the maximum number of leaves to delete, F is the set of locked triplets so far.
if $q < 0$ *or* F *contains conflicting triplets* **then**
  return False;
else if *there are* $ab|c \in F$ *and* $T_i \in \mathcal{T}$ *with* $ac|b \in tr(T_i)$ *or* $bc|a \in tr(T_i)$ **then**
  Branching: If one of the following calls returns True:
  MASTRL$((\mathcal{T} \setminus \{T_i\}) \cup \{T_i - \{a\}\}, q-1, F)$; /* remove a from T_i */
  MASTRL$((\mathcal{T} \setminus \{T_i\}) \cup \{T_i - \{b\}\}, q-1, F)$; /* remove b from T_i */
  MASTRL$((\mathcal{T} \setminus \{T_i\}) \cup \{T_i - \{c\}\}, q-1, F)$; /* remove c from T_i */
  then **return** True, otherwise **return** False;
else if *there are* $a, b, c \in \mathcal{X}$ *such that* $|\{ab|c, ac|b, bc|a\} \cap tr(\mathcal{T})| \geq 2$ **then**
  Branching: If one of the following calls returns True:
  MASTRL$(\mathcal{T}, q, F \cup \{ab|c\})$
  MASTRL$(\mathcal{T}, q, F \cup \{ac|b\})$
  MASTRL$(\mathcal{T}, q, F \cup \{bc|a\})$
  then **return** True, otherwise **return** False;
else if *there is a conflicting set* $\{a, b, c, d\}$ *in* $tr(\mathcal{T}) \cup F$ **then**
  Branching: If one of the following calls returns True:
  MASTRL$(\mathcal{T}, q, F \cup \{ac|b\})$
  MASTRL$(\mathcal{T}, q, F \cup \{bc|a\})$
  MASTRL$(\mathcal{T}, q, F \cup \{bd|c\})$
  MASTRL$(\mathcal{T}, q, F \cup \{ab|d\})$
  then **return** True, otherwise **return** False;
else
  return True ; /* There are no conflicts $\Rightarrow tr(\mathcal{T}) \cup F$ is compatible */
end

Algorithm 1: MASTRL$(\mathcal{T}, q, F)$ — Recursive AST-LR FPT algorithm.

Although Theorem 7 is theoretically interesting as it shows that AST-LR is in FPT with respect to q, the 12^q factor might be too high for practical purposes, motivating the alternative approach below.

4.2 Parameterization by Maximum Distance d

We now describe an algorithm for the `AST-LR-d` problem, running in time $O(c^d d^{3d}(n^3 + tn \log n))$ that, if it exists, finds a solution (where here c is a constant not depending on d nor n). We employ a branch-and-bound strategy. Taking $T = T_1$ as our initial solution, we transform a candidate solution T until we have $d_{LR}(T, T_i) \leq d$ for every input tree T_i.

The type of transformations we use are *leaf prune-and-regraft (LPR)* moves, which provide another way of characterising the distance function d_{LR}. Informally speaking, an LPR move prunes a leaf from a tree and then regrafts it another location. We now give a more formal definition:

Definition 8. *Let T be a tree on label set $\mathcal{X}$. A LPR move on T is a pair (ℓ, e) where $\ell \in \mathcal{X}$ and $e \in \{E(T - \{\ell\}), \perp\}$. Applying (ℓ, e) consists of grafting ℓ on the e edge of $T - \{\ell\}$ if $e \neq \perp$, and above $T - \{\ell\}$ if $e = \perp$.*

An LPR sequence *$L = ((\ell_1, e_1), \ldots, (\ell_k, e_k))$ is an ordered tuple of LPR moves, where for each $i \in [k]$, (ℓ_i, e_i) is an LPR move on the tree obtained after applying the first $i - 1$ LPR moves of L.*

Lemma 9. *Given two trees T_1 and T_2 on label set $\mathcal{X}$, there is a subset $X \subseteq \mathcal{X}$ such that $T_1 - X = T_2 - X$ if and only if there exists an LPR sequence $((x_1, e_1), \ldots, (x_k, e_k))$ turning T_1 into T_2 such that $X = \{x_1, \ldots, x_k\}$. Furthermore, if such a sequence exists then for each $i \in [k]$, there also exists an LPR sequence $L' = ((x'_1, e'_1), \ldots, (x'_k, e'_k))$ turning T_1 into T_2 such that $X = \{x'_1, \ldots, x'_k\}$ and $x'_1 = x_i$.*

Lemma 9 implies that in order for our algorithm to find a solution, it is enough to choose the correct LPR move on T at each stage. In order to get the desired running time, we need to bound the number of possible transformations to try on T.

This can be done as follows. Given a tree T_i with $d_{LR}(T, T_i) > d$, let us call a leaf x *interesting* if there is a solution T^*, and minimal sets $X', X_i \subseteq \mathcal{X}$ of size at most d, such that (a) $T - X' = T^* - X'$, (b) $T_i - X_i = T^* - X_i$, and (c) $x \in X' \setminus X_i$. (Roughly speaking, x is in the 'wrong place' in T but not T_i.)

The following lemma shows that if a solution T^* exists, then T^* can always be reached by moving an interesting leaf at each stage.

Lemma 10. *Suppose that $d < d_{LR}(T_1, T_2) \leq d' + d$ with $d' \leq d$, and that there is a tree T^* and subsets $X_1, X_2 \subseteq \mathcal{X}$ such that $T_1 - X_1 = T^* - X_1$, $T_2 - X_2 = T^* - X_2$ and $|X_1| \leq d', |X_2| \leq d$. Then, there is a minimal label-disagreement X for $\{T_1, T_2\}$ with $|X| \leq d + d'$, and there exists $x \in X$ such that $x \in X_1 \setminus X_2$.*

Moreover, we can in polynomial time construct a set S of size $O(d^2)$ containing all interesting leaves:

Lemma 11. *Suppose that $d_{LR}(T_1, T_2) \leq d$ for some integer d. Then, there is some $S \subseteq \mathcal{X}$ such that $|S| \leq 8d^2$, and for any minimal label-disagreement X for $\{T_1, T_2\}$ with $|X| \leq d$, $X \subseteq S$. Moreover S can be found in time $O(n^2)$.*

The idea behind the proof of Lemma 11 is as follows: In polynomial time, we can find a set $X' \subseteq \mathcal{X}$ for which $T_1 - X' = T_2 - X'$. Letting X_1 and X_2 be disjoint copies of X, it is easy to construct a tree T_J with label set $(\mathcal{X} \setminus X') \cup X_1 \cup X_2$, such that $T_J - X_2 = T_1$ and $T_J - X_1 = T_2$. Such a tree therefore represents the structure of T_1 and T_2 at the same time. Moreover, by letting T^* be the subtree of T_J spanning $X_1 \cup X_2$, we have that T_J can be derived from T^* by grafting trees (on subsets of $\mathcal{X} \setminus X'$) onto edges of T_J. We call these subtrees *dangling clades*.

It can be shown that for any dangling clade, any minimal label-disagreement for $\{T_1, T_2\}$ either contains all labels from that clade or contains none of them. Moreover, if there are multiple dangling clades grafted onto the same edge of T^*, then a minimal label-disagreement for $\{T_1, T_2\}$ either contains the labels of every such dangling clade, or every such dangling clade except one, or none of them.

As a result, we can construct our set S by taking X' together with any combination of clades as described above that has total size at most d. It can be shown that S in fact has at most $2d$ labels for each edge of T^*, and as T^* has $O(d)$ edges, we get the desired bound on $|S|$.

The last ingredient needed for Theorem 13 is Lemma 12, which shows that if a leaf x of T_1 as described in Lemma 10 has to be moved, then there are not too many ways to regraft it in order to get closer to T^*. This gives us a bound of $O(d^3)$ on the number of branches at each step of our search tree, which in turn implies that there are at most $O(c^d d^{3d})$ steps.

Lemma 12. *Suppose that $d < d_{LR}(T_1, T_2) \leq d' + d$ with $d' \leq d$, and that there are $X_1, X_2 \subseteq \mathcal{X}$, and a tree T^* such that $T_1 - X_1 = T^* - X_1, T_2 - X_2 = T^* - X_2, |X_1| \leq d', |X_2| \leq d$, and let $x \in X_1 \setminus X_2$. Then, there is a set P of trees on label set $\mathcal{X}$ that satisfies the following conditions:*

- *for any tree T' such that $d_{LR}(T', T^*) < d_{LR}(T_1, T^*)$ and T' can be obtained from T_1 by pruning a leaf x and regrafting it, $T' \in P$;*
- *$|P| \leq 18(d + d') + 8$;*
- *P can be found in time $O(n(\log n + 18(d + d') + 8))$.*

The idea behind the proof of Lemma 12 is as follows: by looking at a subtree common to T_1 and T_2, we can identify the location that T_2 "wants" x to be positioned. This may not be the correct position for x, but we can show that if x is moved too far from this position, we will create a large number of conflicting triplets between T_2 and the solution T^*. As a result, we can create all trees in P by removing x from T_1 and grafting it on one of a limited number of edges.

Putting everything together, we have the procedure outlined in Algorithm 2. (In this algorithm, the subroutines DISAGREEMENT-KERNEL and CANDIDATE-TREES refer to the agorithms described in Lemmas 11 and 12, respectively.) Analysing this algorithm gives the desired running time.

Theorem 13. *`AST-LR-d` can be solved in time $O(c^d d^{3d}(n^3 + tn \log n))$, where c is a constant not depending on d or n.*

Data: $\mathcal{T}$ is the set of input trees (represented as a sequence to distinguish T_1 from the other trees), d is the maximum number of leaves we can remove in a tree, d' is the maximum number of leaves we can move in T_1, which should be initially set to d.
if $d_{LR}(T_1, T_i) \leq d$ *for each* $T_i \in \mathcal{T}$ **then**
  return T_1;
else if *there is some* $T_i \in \mathcal{T}$ *such that* $d_{LR}(T_1, T_i) > d' + d$ **then**
  return False ; `/* handles the` $d' \leq 0$ `case */`
else
  `/* Here we 'guess' a leaf prune-and-regraft move on` T_1 `*/`
  Choose $T_i \in \mathcal{T}$ such that $d_{LR}(T_1, T_i) > d$;
  Set $S =$ DISAGREEMENT-KERNEL$(d + d', T_1, T_i)$;
  for $x \in S$ **do**
    `/* We are 'guessing' that` x `should go where` T_i `wants it. */`
    Set $P =$ CANDIDATE-TREES(T_1, T_i, x, d, d');
    $T^* = False$;
    for $T \in P$ **do**
      $T' =$ MASTRL−DISTANCE$((T, T_2, \ldots, T_t), d, d' - 1)$;
      If T' is not $False$, let $T^* := T'$;
    end
    return T^*;
  end
end

Algorithm 2: MASTRL−DISTANCE$(\mathcal{T} = (T_1, T_2, \ldots, T_t), d, d')$ — FPT algorithm for parameter d.

5 Conclusion

To conclude, we introduced a new supertree/consensus problem, based on a simple combinatorial operator acting on trees, the Leaf-Removal. We showed that, although this supertree problem is NP-hard, it admits interesting tractability results, that compare well with existing algorithms. Future research should explore if various simple combinatorial operators, that individually define relatively tractable supertree problems (for example LR and EC) can be combined into a unified supertree problem while maintaining approximability and fixed-parameter tractability.

Acknowledgements

MJ was partially supported by Labex NUMEV (ANR-10-LABX-20) and Vidi grant 639.072.602 from The Netherlands Organization for Scientific Research (NWO). CC was supported by NSERC Discovery Grant 249834. CS was partially

supported by the French Agence Nationale de la Recherche Investissements dAvenir/Bioinformatique (ANR-10-BINF-01-01, ANR-10-BINF-01-02, Ancestrome). ML was supported by NSERC PDF Grant. MW was partially supported by the Institut de Biologie Computationnelle (IBC).

References

1. Aberer, A.J., Krompass, D., Stamatakis, A.: Pruning rogue taxa improves phylogenetic accuracy: An efficient algorithm and webservice. Systematic Biology 62(1), 162–166 (2013), + http://dx.doi.org/10.1093/sysbio/sys078
2. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. SIAM J. Comput. 26, 1656–1669 (1997), http://dx.doi.org/10.1137/S0097539794269461
3. Bryant, D.: Building trees, hunting for trees, and comparing trees. Ph.D. thesis, Bryant University (1997)
4. Bryant, D., McKenzie, A., Steel, M.: The size of a maximum agreement subtree for random binary trees. Dimacs Series in Discrete Mathematics and Theoretical Computer Science 61, 55–66 (2003)
5. Byrka, J., Guillemot, S., Jansson, J.: New results on optimizing rooted triplets consistency. Discrete Appl. Math. 158, 1136–1147 (2010), http://dx.doi.org/10.1016/j.dam.2010.03.004
6. Chauve, C., Jones, M., Lafond, M., Scornavacca, C., Weller, M.: Constructing a consensus phylogeny from a leaf-removal distance. http://arxiv.org/abs/1705.05295
7. Chester, A., Dondi, R., Wirth, A.: Resolving Rooted Triplet Inconsistency by Dissolving Multigraphs, pp. 260–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
8. Cole, R., Farach-Colton, M., Hariharan, R., Przytycka, T.M., Thorup, M.: An $O(n\log\ n)$ algorithm for the maximum agreement subtree problem for binary trees. SIAM J. Comput. 30, 1385–1404 (2000), http://dx.doi.org/10.1137/S0097539796313477
9. Deng, Y., Fernández-Baca, D.: Fast Compatibility Testing for Rooted Phylogenetic Trees. In: Combinatorial Pattern Matching 2016. LIPIcs. Leibniz Int. Proc. Inform., vol. 54, pp. 12:1–12:12 (2016), http://drops.dagstuhl.de/opus/volltexte/2016/6088
10. Fernández-Baca, D., Guillemot, S., Shutters, B., Vakati, S.: Fixed-parameter algorithms for finding agreement supertrees. SIAM J. Comput. 44, 384–410 (2015), http://dx.doi.org/10.1137/120897559
11. Guillemot, S., Mnich, M.: Kernel and fast algorithm for dense triplet inconsistency. Theoret. Comput. Sci. 494, 134–143 (2013), http://dx.doi.org/10.1016/j.tcs.2012.12.032
12. Gusfield, D.: Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge university press (1997)
13. Hellmuth, M., Wieseke, N., Lechner, M., Lenhof, H.P., Middendorf, M., Stadler, P.F.: Phylogenomics with paralogs. Proc. Natl. Acad. Sci. USA 112, 2058–2063 (2015), http://dx.doi.org/10.1073/pnas.1412770112
14. Jarvis, E.D., et al.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science 346, 1320–1331 (2014), http://dx.doi.org/10.1126/science.1253451

15. Scornavacca, C., Galtier, N.: Incomplete lineage sorting in mammalian phylogenomics. Sys. Biol. 66, 112–120 (2017), http://dx.doi.org/10.1093/sysbio/syw082
16. Scornavacca, C., Jacox, E., Szollösi, G.J.: Joint amalgamation of most parsimonious reconciled gene trees. Bioinformatics 31, 841–848 (2015), http://dx.doi.org/10.1093/bioinformatics/btu728
17. Szollösi, G.J., Boussau, B., Abby, S.S., Tannier, E., Daubin, V.: Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. Proc. Natl. Acad. Sci. USA 109, 17513–17518 (2012), http://dx.doi.org/10.1073/pnas.1202997109
18. Vachaspati, P., Warnow, T.: FastRFS: fast and accurate robinson-foulds supertrees using constrained exact optimization. Bioinformatics 33, 631–639 (2017), http://dx.doi.org/10.1093/bioinformatics/btw600
19. Whidden, C., Zeh, N., Beiko, R.G.: Supertrees based on the subtree prune-and-regraft distance. Sys. Biol. 63, 566–581 (2014), http://dx.doi.org/10.1093/sysbio/syu023