Annals of Operations Research (2022) 309:425–451
https://doi.org/10.1007/s10479-021-04352-1

ORIGINAL - OR MODELING/CASE STUDY

Reflections on kernelizing and computing unrooted agreement forests

Rim van Wersch[1] · **Steven Kelk**[1] · **Simone Linz**[2] · **Georgios Stamoulis**[1]

Accepted: 30 September 2021 / Published online: 19 November 2021

Abstract
Phylogenetic trees are leaf-labelled trees used to model the evolution of species. Here we explore the practical impact of kernelization (i.e. data reduction) on the NP-hard problem of computing the TBR distance between two unrooted binary phylogenetic trees. This problem is better-known in the literature as the maximum agreement forest problem, where the goal is to partition the two trees into a minimum number of common, non-overlapping subtrees. We have implemented two well-known reduction rules, the subtree and chain reduction, and five more recent, theoretically stronger reduction rules, and compare the reduction achieved with and without the stronger rules. We find that the new rules yield smaller reduced instances and thus have clear practical added value. In many cases they also cause the TBR distance to decrease in a controlled fashion, which can further facilitate solving the problem in practice. Next, we compare the achieved reduction to the known worst-case theoretical bounds of $15k-9$ and $11k-9$ respectively, on the number of leaves of the two reduced trees, where k is the TBR distance, observing in both cases a far larger reduction in practice. As a by-product of our experimental framework we obtain a number of new insights into the actual computation of TBR distance. We find, for example, that very strong lower bounds on TBR distance can be obtained efficiently by randomly sampling certain carefully constructed partitions of the leaf labels, and identify instances which seem particularly challenging to solve exactly. The reduction rules have been implemented within our new solver *Tubro* which combines kernelization with an Integer Linear Programming (ILP) approach. Tubro also incorporates a number of additional features, such as a cluster reduction and a practical upper-bounding heuristic, and it can leverage combinatorial insights emerging from the proofs of correctness of the reduction rules to simplify the ILP.

Keywords Phylogenetics · Agreement forest · TBR distance · Kernelization · Fixed parameter tractability

1 Introduction

The central challenge of phylogenetics, which is the study of phylogenetic (evolutionary) trees, is to infer a tree whose leaves are bijectively labeled by a set X of species and which accurately represents the evolutionary events that gave rise to X. There are many methods

Extended author information available on the last page of the article

to construct phylogenetic trees; we refer to a standard text such as Felsenstein (2004) for an overview. The complexity of this problem stems from the fact that we typically only have indirect data available, such as DNA sequences of a set X of species. It is quite common for different methods to construct different trees, or the same method to construct different trees depending on which part of a genome the DNA data is extracted from Huson et al. (2011), Richards et al. (2018), Yoshida et al. (2019). In such cases we might wish to formally quantify the *dissimilarity* of these trees, and this has fuelled research into different distance measures on pairs of phylogenetic trees (Kuhner and Yamato 2015). Such distances also help us to understand (i) the space of all phylogenetic trees on a fixed number of leaves (John 2017), (ii) the construction of phylogenetic supertrees (Whidden et al. 2014), which produces a single phylogenetic tree summarizing multiple trees; and (iii) phylogenetic networks (Huson et al. 2011), which generalize phylogenetic trees to graphs.

In this article we focus on the following problem: given two unrooted binary phylogenetic trees T and T', both on leaf set X, what is the minimum number of Tree Bisection and Reconnection (TBR) moves required to turn T into T'? We defer formal definitions to Sect. 2.1, but in essence a TBR move consists of deleting an edge of a tree and then introducing a new edge to reconnect the two resulting components back together. This distance measure, denoted d_{TBR}, is NP-hard to compute (Allen and Steel 2001; Hein et al. 1996). Computation of d_{TBR} is essentially equivalent to the well-known problem of computing a *maximum agreement forest* of T and T'. Such a forest is a partition of X into the minimum number of blocks so that each block induces a subtree in T that is isomorphic (up to subdivision) to that induced in T', and no two blocks induce overlapping subtrees in T and T'. This minimum number, d_{MAF}, is equal to $d_{\mathrm{TBR}} + 1$ (Allen and Steel 2001). Due to this equivalence, everything in this article that applies to d_{TBR} also applies to the computation of d_{MAF}.

In the last few years there has been quite some research into the *fixed parameter tractability* of computing d_{TBR}. Fixed parameter tractable (FPT) algorithms are those with running time $O(h(k) \cdot \mathrm{poly}(n))$ where n is the size of the input and h is a computable function that depends only on some well-chosen parameter k (Downey and Fellows 2013; Cygan et al. 2015). Such algorithms have the potential to run quickly when k is small, even if n is large. In any case, they run more quickly than algorithms with running times of the form $O(n^{h(k)})$. A common technique for developing FPT algorithms is to navigate a computational search tree such that the number of vertices in the tree is bounded by a function of k. Using this branching technique, running times of $O(4^k \cdot \mathrm{poly}(|X|))$ (Whidden et al. 2013) and later $O(3^k \cdot \mathrm{poly}(|X|))$ (Chen et al. 2015) have been obtained for the computation of d_{TBR}, where $k = d_{\mathrm{TBR}}$. Another main technique is *kernelization* (Fomin et al. 2019), which is the focus of this article. The goal here is to apply polynomial-time pre-processing reduction rules such that the reduced instance—the *kernel*—has size bounded by a function of k. Such a kernel can then be solved by an exact exponential-time algorithm, which ultimately also yields a running time of the form $O(h(k) \cdot \mathrm{poly}(n))$. For the computation of d_{TBR}, a kernel of size $28k$ was established in 2001; recently this was reduced to $15k - 9$, and later $11k - 9$, by two of the present authors Kelk and Linz (2019, 2020). The $28k$ and $15k - 9$ results are based on two well-known reduction rules, the *subtree* and *chain* reduction rules, while the $11k - 9$ result was obtained by adding five new reduction rules to this repertoire. Roughly speaking, the latter five reduction rules target short chains that cannot be (further) reduced by the chain reduction rule and, so, the $11k - 9$ result is based on repeated applications of all seven reduction rules. The $15k - 9$ and $11k - 9$ bounds are both tight under their respective sets of reduction rules.

In this article we adopt an experimental approach to answering the following question: do the new reduction rules from Kelk and Linz (2020) produce smaller kernels *in practice*

than, say, when only the subtree and chain reductions are applied? This mirrors several recent articles in the algorithmic graph theory literature where the practical effectiveness of kernelization has also been analyzed (Fellows et al. 2018; Ferizovic et al. 2020; Henzinger et al. 2020; Mertzios et al. 2020; Alber et al. 2006). The question is relevant, since earlier studies of kernelization in phylogenetics have noted that, despite its theoretical importance, in an empirical setting the chain reduction seems to have very limited effect compared to the subtree reduction (Hickey et al. 2008; van Iersel et al. 2016). In this sense, it is natural to ask whether the five new reduction rules have any real added value in practice. Happily, our experiments indicate that the answer is *yes*; across a range of experimental settings, the average percentage of species contained in X that survive after application of the entire ensemble of reduction rules, is substantially smaller than when only the subtree and chain reductions are applied. We also explore whether the reduction achieved in practice is, expressed as a function of k, significantly better than the theoretical worst-case bound of $11k-9$. Again, we answer this affirmatively. Our experiments show that kernels of size $6k$ or lower are obtained for 93% of the tree pairs that we used in our experimental study. We also derive insights into how frequently the phenomenon of *parameter reduction* occurs: this is when a reduction rule is triggered that does not preserve d_{TBR} but instead reduces it in some predictable and well-understood fashion. Parameter reduction is particularly helpful if the kernel is subsequently solved by an FPT algorithm whose running time is exponentially dependent on d_{TBR}.

Our experimental framework also yields a number of new insights concerning the actual *computation* of d_{TBR} as opposed to only reducing the problem in size. To compute the empirical kernel size mentioned in the previous paragraph, it is necessary to know d_{TBR}, or a good lower bound on d_{TBR}. To that end our experiments deploy the only existing, publicly-available TBR solver uSPR (Whidden and Matsen 2018) to compute d_{TBR}, where possible, which is not always the case. In this way we build up a picture of which instances are particularly challenging to compute in practice. For instances where computation of d_{TBR} is too challenging for uSPR, we use a novel lower bound on d_{TBR} based on randomly sampling certain types of so-called *convex characters*, which are specially constructed partitions of X (Semple and Steel 2003). This turns out to be *extremely* effective, often yielding lower bounds within a few seconds that are very close to d_{TBR}. Arguably, alongside our study of kernel size, the discovery of this bound is the most important contribution of this paper.

A final contribution of this article is our new solver *Tubro*. This was initially intended simply as a vehicle for the new reduction rules from Kelk and Linz (2020). However, during development we reasoned that it would be useful for *Tubro* to incorporate its own solver. To this end, we give a new Integer Linear Programming (ILP) formulation for computation of d_{TBR}, and prove its correctness. Tubro feeds this ILP to the powerful solver (Gurobi 2020), strengthening the formulation with heuristically computed upper bounds and the aforementioned lower bound. We note that Tubro is capable of solving a number of comparatively small instances that are out-of-range for uSPR in our experiments. Tubro additionally incorporates a number of extra features that are possibly of independent interest, such as a *cluster* reduction (Bordewich et al. 2017; Li and Zeh 2017). It is also able to significantly reduce the number of variables in the ILP by specifying that even in fully kernelized instances certain chains should never be cut, leveraging an insight used in the proofs of correctness in Kelk and Linz (2020). An executable version of Tubro is available upon request.

The paper is organized as follows. The next section contains some preliminaries that are used throughout the paper and details the seven reduction rules that are underlying the kernelization experiments presented here. In Sect. 3, we introduce the maximum parsimony distance between two unrooted binary phylogenetic trees and the associated sampling strategy to obtain a lower bound on the TBR distance. The following three sections describe our

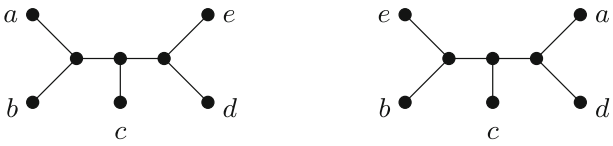


Fig. 1 Two unrooted binary phylogenetic trees on $X = \{a, b, c, d, e\}$

experiments. We first present the experimental setup in Sect. 4 and, subsequently, summarize and analyse the results in Sect. 5. A high-level discussion follows in Sect. 6, where we reflect on what our experiments tell us about which TBR instances are easy, and difficult, to solve, and why this might be. In our conclusion Sect. 7 we make a number of suggestions for further research. Technical details of Tubro are deferred to the appendix, which establishes correctness of the ILP that Tubro is based on and gives details of the various options that can be switched on or off when running Tubro, such as chain preservation and cluster reduction.

2 Preliminaries

2.1 Definitions

Our notation closely follows (Kelk and Linz 2020). Throughout this paper, X denotes a finite set of *taxa*. An *unrooted binary phylogenetic tree* T on X is a simple, connected, and undirected tree whose leaves are bijectively labeled with X and whose other vertices all have degree 3. See Fig. 1 for an example of two unrooted binary phylogenetic trees on $X = \{a, b, c, d, e\}$. For simplicity and since most phylogenetic trees in this paper are unrooted and binary, we refer to an unrooted binary phylogenetic trees as a *phylogenetic tree*. If a definition or statement applies to all unrooted phylogenetic trees, regardless of whether they are binary or not, we make this explicit. Two leaves, say a and b, of T are called a *cherry* $\{a, b\}$ of T if they are adjacent to a common vertex. For $X' \subseteq X$, we write $T[X']$ to denote the unique, minimal subtree of T that connects all elements in X'. For brevity we call $T[X']$ the *embedding* of X' in T. Furthermore, we refer to the phylogenetic tree on X' obtained from $T[X']$ by suppressing non-root degree-2 vertices as the *restriction of* T *to* X' and we denote this by $T|X'$.

Tree bisection and reconnection Let T be a phylogenetic tree on X. Apply the following three-step operation to T:

1. Delete an edge in T and suppress any resulting degree-2 vertex. Let T_1 and T_2 be the two resulting phylogenetic trees.
2. If T_1 (resp. T_2) has at least one edge, subdivide an edge in T_1 (resp. T_2) with a new vertex v_1 (resp. v_2) and otherwise set v_1 (resp. v_2) to be the single isolated vertex of T_1 (resp. T_2).
3. Add a new edge $\{v_1, v_2\}$ to obtain a new phylogenetic tree T' on X.

We say that T' has been obtained from T by a single *tree bisection and reconnection (TBR) operation* (or, *TBR move*). Furthermore, we define the TBR *distance* between two phylogenetic trees T and T' on X, denoted by $d_{\mathrm{TBR}}(T, T')$, to be the minimum number of TBR operations that are required to transform T into T'. To illustrate, the trees T and T' in Fig. 2 have a TBR distance of 1. It is well known that d_{TBR} is a metric (Allen and Steel 2001). By

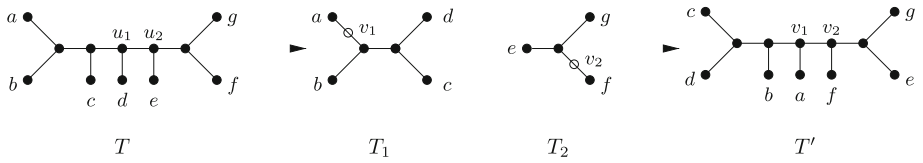


Fig. 2 A single TBR operation that transforms T into T'. First, T_1 and T_2 are obtained from T by deleting the edge $\{u_1, u_2\}$ in T. Second, T' is obtained from T_1 and T_2 by subdividing an edge in both trees as indicated by the open circles v_1 and v_2 and adding a new edge $\{v_1, v_2\}$

building on an earlier result by Hein et al. (1996, Theorem 8), Allen and Steel (2001) showed that computing the TBR distance is an NP-hard problem.

Agreement forests Let T and T' be two phylogenetic trees on X. Furthermore, let $F = \{B_0, B_1, B_2, \ldots, B_k\}$ be a partition of X, where each block B_i with $i \in \{0, 1, 2, \ldots, k\}$ is referred to as a *component* of F. We say that F is an *agreement forest* for T and T' if the following conditions hold.

(1) For each $i \in \{0, 1, 2, \ldots, k\}$, we have $T|B_i = T'|B_i$.
(2) For each pair $i, j \in \{0, 1, 2, \ldots, k\}$ with $i \neq j$, we have that $T[B_i]$ and $T[B_j]$ are vertex-disjoint in T, and $T'[B_i]$ and $T'[B_j]$ are vertex-disjoint in T'.

Let $F = \{B_0, B_1, B_2, \ldots, B_k\}$ be an agreement forest for T and T'. The *size* of F is simply its number of components; i.e. $k + 1$. Moreover, an agreement forest with the minimum number of components (over all agreement forests for T and T') is called a *maximum agreement forest* (*MAF*) for T and T'. The number of components of a maximum agreement forest for T and T' is denoted by $d_{\mathrm{MAF}}(T, T')$. The following theorem is well known.

Theorem 1 (Allen and Steel 2001, Theorem 2.13) *Let T and T' be two phylogenetic trees on X. Then*

$$d_{\mathrm{TBR}}(T, T') = d_{\mathrm{MAF}}(T, T') - 1.$$

A maximum agreement forest for the trees T and T' shown in Fig. 2, which have TBR distance 1, therefore contains two components. $F = \{\{a, b, c, d\}, \{e, f, g\}\}$ is an example of such a forest (in fact, here it is the only maximum agreement forest).

We conclude this section with a number of algorithmic definitions. A *parameterized problem* is a problem for which the inputs are of the form (x, k), where k is a non-negative integer, called the *parameter*. A parameterized problem is *fixed-parameter tractable* (FPT) if there exists an algorithm that solves[1] any instance (x, k) in $h(k) \cdot |x|^{O(1)}$ time, where $h(\cdot)$ is a computable function depending only on k. A parameterized problem has a *kernel* of size $g(k)$, where $g(\cdot)$ is a computable function depending only on k, if there exists a polynomial time algorithm transforming any instance (x, k) into an equivalent instance (x', k'), with $|x'|, k' \leq g(k)$. Here, *equivalent* means that (x, k) is a yes-instance if and only if (x', k') is a yes-instance. If $g(k)$ is a polynomial in k then we call this a *polynomial kernel*; if $g(k) = O(k)$ then it is a *linear kernel*. A well-known theorem from parameterized complexity states that a parameterized problem is fixed-parameter tractable if and only if it has a (not necessarily polynomial) kernel. For more background information on fixed parameter tractability and

[1] Note that the formalism described here actually concerns *decision* (i.e. yes/no) problems, which in the context of the current article is most naturally "Is $d_{\mathrm{TBR}} \leq k$?". An FPT algorithm for answering this question can easily be transformed into an algorithm for computing d_{TBR} with similar asymptotic time complexity by increasing k incrementally from 0 until a yes-answer is obtained.

kernelization, we refer the reader to standard texts such as Cygan et al. (2015), Downey and Fellows (2013), Fomin et al. (2019).

In this article, we take d_{TBR} as the parameter k and take $|X|$, the number of leaves, as the size of the instance $|x|$. The reduction rules described in the following section produce a linear kernel and run in poly($|X|$) time.

2.2 Description of the subtree, chain and other reduction rules

In this section we describe the existing reduction rules—seven in total—that will be analyzed in this article.

Let T and T' be two phylogenetic trees on X. We say that a subtree of T is *pendant* if it can be detached from T by deleting a single edge. For reasons of simplicity, we assume for the remainder of this paragraph that $|X| \geq 4$. Note that no generality is lost because $d_{\mathrm{TBR}}(T, T') = 0$ if $|X| < 4$. For $n \geq 2$, let $C = (\ell_1, \ell_2 \ldots, \ell_n)$ be a sequence of distinct taxa in X. For each $i \in \{1, 2 \ldots, n\}$, let p_i denote the unique parent of ℓ_i in T. We call C an n-chain of T if there exists a walk $p_1, p_2, \ldots, p_n$ in T and the elements $p_2, p_3, \ldots, p_{n-1}$ are all pairwise distinct. Note that ℓ_1 and ℓ_2 may have a common parent or ℓ_{n-1} and ℓ_n may have a common parent. Furthermore, if $p_1 = p_2$ or $p_{n-1} = p_n$ holds, then C is pendant in T. To ease reading, we sometimes write C to denote the set $\{\ell_1, \ell_2, \ldots, \ell_n\}$. It will always be clear from the context whether C refers to the associated sequence or set of taxa. If a pendant subtree S (resp. an n-chain C) exists in T and T', we say that S (resp. C) is a *common* subtree (resp. chain) of T and T'.

We are now in a position to state all seven reduction rules. Let T and T' be two phylogenetic trees on X. We already note here that an application of Reduction 1, 2, 6, or 7 results in two new trees whose TBR distance is the same as that of T and T', and an application of Reduction 3, 4, or 5, results in two new trees whose TBR distance is one less than that of T and T'.

Reduction 1 (Allen and Steel 2001) If T and T' have a maximal common pendant subtree S with at least two leaves, then reduce T and T' to T_r and T'_r, respectively, by replacing S with a single leaf with a new label.

Reduction 2 (Allen and Steel 2001) If T and T' have a maximal common n-chain $C = (\ell_1, \ell_2, \ldots, \ell_n)$ with $n \geq 4$, then reduce T and T' to $T_r = T|X\backslash\{\ell_4, \ell_5, \ldots, \ell_n\}$ and $T'_r = T'|X\backslash\{\ell_4, \ell_5, \ldots, \ell_n\}$, respectively.

Reduction 3 (Kelk and Linz 2020) If T and T' have a common 3-chain $C = (\ell_1, \ell_2, \ell_3)$ such that $\{\ell_1, \ell_2\}$ is a cherry in T and $\{\ell_2, \ell_3\}$ is a cherry in T', then reduce T and T' to $T_r = T|X\backslash C$ and $T'_r = T'|X\backslash C$, respectively.

Reduction 4 (Kelk and Linz 2020) If T and T' have a common 3-chain $C = (\ell_1, \ell_2, \ell_3)$ such that $\{\ell_2, \ell_3\}$ is a cherry in T and $\{\ell_3, x\}$ is a cherry in T' with $x \in X\backslash C$, then reduce T and T' to $T_r = T|X\backslash\{x\}$ and $T'_r = T'|X\backslash\{x\}$, respectively.

Reduction 5 (Kelk and Linz 2020) If T and T' have two common 2-chains $C_1 = (\ell_1, \ell_2)$ and $C_2 = (\ell_3, \ell_4)$ such that T has cherries $\{\ell_2, x\}$ and $\{\ell_3, \ell_4\}$, and T' has cherries $\{\ell_1, \ell_2\}$ and $\{\ell_4, x\}$ with $x \in X\backslash(C_1 \cup C_2)$, then reduce T and T' to $T_r = T|X\backslash\{x\}$ and $T'_r = T'|X\backslash\{x\}$, respectively.

Reduction 6 (Kelk and Linz 2020) If T and T' have two common 3-chains $C_1 = (\ell_1, \ell_2, \ell_3)$ and $C_2 = (\ell_4, \ell_5, \ell_6)$ such that T has cherries $\{\ell_2, \ell_3\}$ and $\{\ell_4, \ell_5\}$, and $(\ell_1, \ell_2, \ldots, \ell_6)$ is a 6-chain of T', then reduce T and T' to $T_r = T|X\backslash\{\ell_4, \ell_5\}$ and $T'_r = T'|X\backslash\{\ell_4, \ell_5\}$, respectively.

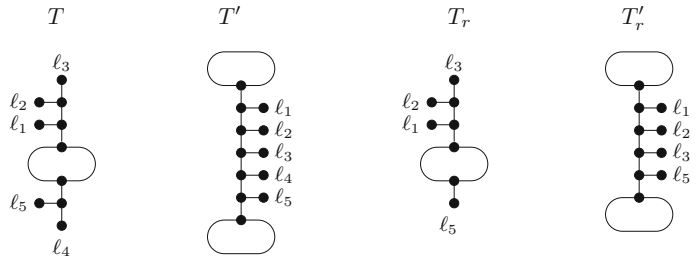


Fig. 3 An example of Reduction 7. Ovals indicate subtrees

Reduction 7 (Kelk and Linz 2020) If T and T' have common chains $C_1 = (\ell_1, \ell_2, \ell_3)$ and $C_2 = (\ell_4, \ell_5)$ such that T has cherries $\{\ell_2, \ell_3\}$ and $\{\ell_4, \ell_5\}$, and $(\ell_1, \ell_2, \ldots, \ell_5)$ is a 5-chain of T', then reduce T and T' to $T_r = T|X\backslash\{\ell_4\}$ and $T'_r = T'|X\backslash\{\ell_4\}$, respectively.

An example of Reduction 7 is illustrated in Fig. 3. Reduction 1 is known as *subtree reduction* while Reduction 2 is known as *chain reduction* in the literature (for example, see Allen and Steel 2001). Reductions 3–7 assume that Reductions 1 and 2 have already been applied to exhaustion.

The following two lemmas summarize results established in Allen and Steel (2001), Kelk and Linz (2019, 2020).

Lemma 1 *Let T and T' be two phylogenetic trees on X. If T_r and T'_r are two phylogenetic trees obtained from T and T', respectively, by a single application of Reduction 1, 2, 6, or 7, then $d_{\mathrm{TBR}}(T, T') = d_{\mathrm{TBR}}(T_r, T'_r)$. Moreover, if T_r and T'_r are two trees obtained from T and T', respectively, by a single application of Reduction 3, 4, or 5, then $d_{\mathrm{TBR}}(T, T') - 1 = d_{\mathrm{TBR}}(T_r, T'_r)$.*

Lemma 2 *Let S and S' be two phylogenetic trees on X that cannot be reduced by Reduction 1 or 2, and let T and T' be two phylogenetic trees on Y that cannot be reduced by any of the seven reductions. If $d_{\mathrm{TBR}}(S, S') \geq 2$, then $|X| \leq 15d_{\mathrm{TBR}}(S, S') - 9$. Furthermore, if $d_{\mathrm{TBR}}(T, T') \geq 2$, then $|Y| \leq 11d_{\mathrm{TBR}}(T, T') - 9$.*

Note that an application of Reductions 3, 4, or 5 triggers a *parameter reduction*, whereby the TBR distance is reduced. In these cases, an element of X is located which definitely comprises a singleton component in some maximum agreement forest, and whose deletion thus lowers the TBR distance by 1. Reductions 1, 2, 6 and 7, on the other hand, preserve TBR distance. Reductions 6 and 7 work by truncating short chains, i.e. chains which escape Reduction 2, to be even shorter. It was noted in Kelk and Linz (2020) that an application of Reduction 4 or 5 immediately triggers an application of Reduction 1.

3 Maximum Parsimony distance and a new approach to computing lower bounds on TBR distance

Throughout this section, an unrooted phylogenetic tree T is not necessarily binary, i.e. each internal vertex of T has degree at least 3. A *character* f on X is a function $f : X \to C$, where $C = \{c_1, c_2, \ldots, c_r\}$ is a set of *character states* for some positive integer r. Let T be an unrooted phylogenetic tree on X with vertex set V, and let f be a character on X whose set of

character states is C. An *extension* g of f to V is a function $g : V \to C$ such that $g(\ell) = f(\ell)$ for each $\ell \in X$. Given an extension g of f, let $l_g(T)$ denote the number of edges $\{u, v\}$ in T such that $g(u) \neq g(v)$. Then the *parsimony score* of f on T, denoted by $l_f(T)$, is obtained by minimizing $l_g(T)$ over all possible extensions g of f; this score can easily be computed in polynomial time (Fitch 1971). Following standard terminology in phylogenetics (Semple and Steel 2003), we say that a character f is *convex* on T if $l_f(T) = r - 1$. Equivalently, a convex character is a character where the r minimal spanning trees induced by the r states of the character are vertex disjoint.

For two unrooted phylogenetic trees T and T' on X, the *maximum parsimony distance* d_{MP} (Fischer and Kelk 2016; Moulton and Wu 2015) is defined as

$$d_{\mathrm{MP}}(T, T') = \max_f |l_f(T) - l_f(T')|.$$

It is known that at least one such maximizing f has the following two properties: (i) it is convex on at least one of T and T', (ii) each state in f occurs on at least 2 taxa (Kelk and Fischer 2017).

It has been noted several times in the literature that d_{MP}, which is itself NP-hard to compute, is a lower bound on d_{TBR} (Fischer and Kelk 2016; Moulton and Wu 2015). Experiments in Kelk et al. (2016) on very small trees (up to 25 taxa), in which d_{MP} was computed exactly using the exponential-time algorithm from Kelk and Stamoulis (2017), suggest that d_{MP} is often very close to d_{TBR}. Inspired by this, we here propose a new scaleable strategy for computing good lower bounds on d_{TBR}. Rather than computing d_{MP} exactly, which is too time-intensive, we leverage the fact that *every* (convex) character f gives a lower bound on d_{MP} and thus on d_{TBR}. By sampling many such characters f, and selecting the one that maximizes $|l_f(T) - l_f(T')|$ we expect to achieve a strong lower bound on d_{MP}. This sampling strategy is made possible by Kelk and Stamoulis (2017, Corollary 5) which states that for a given unrooted phylogenetic tree T, a character that is convex on T whereby each state occurs on at least $p \geq 2$ taxa (p constant) can be sampled uniformly at random in linear time and space; let us call such characters *eligible*. The overall strategy, therefore, is to randomly choose between T and T', and then to uniformly at random select an eligible character f (taking $p = 2$), repeating this as often as desired, and at the end returning the largest value of $|l_f(T) - l_f(T')|$ observed. We henceforth call this the d_{MP} *lower bound sampling strategy* or simply d_{MP} *lower bound* if no confusion arises from the context and denote this bound by d^{ℓ}_{MP}.

4 Experimental framework

4.1 uSPR

To compute TBR distances in our experiments we mainly used the solver uSPR that is described in Whidden and Matsen (2018) and available from https://github.com/cwhidden/uspr, version 1.0.1. This solver is designed to compute the so-called unrooted subtree prune and regraft distance between two phylogenetic trees T and T' which is always at least as large as $d_{\mathrm{TBR}}(T, T')$. As one of its subroutines, uSPR incorporates an exact TBR solver, which can be invoked independently (using the `-tbr` switch). The TBR solver uses iterative deepening. More precisely, for two phylogenetic trees T and T' on X, it starts from a polynomial-time computed lower bound k' and repeatedly asks "Is $d_{\mathrm{TBR}}(T, T') \leq k'$?" for increasing values of k' until the question is answered positively. According to Whidden and Matsen

(2018) the TBR solver of uSPR incorporates a number of the enhanced branching cases described in Chen's algorithm (Chen et al. 2015), which computes TBR distance in time $O(3^{d_{\text{TBR}}} \cdot \text{poly}(|X|))$.

4.2 Tubro

The polynomial-time reduction rules described in Sect. 2.2 have been implemented in our package *Tubro*. Tubro's main role in our experiments is to apply these reduction rules. However, it also has a secondary role. Specifically, Tubro can compute TBR distance using Integer Linear Programming (ILP). As explained in the next section, we use Tubro in an attempt to compute TBR distances for those pairs of phylogenetic trees that are out of range of uSPR. Given that the ILP formulation has $O(|X|^4)$ constraints, however, Tubro is limited to pairs of phylogenetic trees on X which, after reduction, have (roughly) $|X| \leq 70$. Tubro has many other features; we defer a full description of the package to the appendix.

4.3 High-level description of the experiments

All experiments were conducted on the Windows Subsystem for Linux (WSL) [Ubuntu 16.04.6 LTS], running under Windows 10, on a 64-bit HP Envy Laptop 13-ad0xx (quad-core i7-7500 @ 2.7 GHz), with 8 Gb of memory. In experiments such as this, which mainly run in memory and whereby disk accesses are limited, WSL has comparable performance to a native Linux system[2].

Our main dataset comprises 735 *tree pairs*, where each such pair (T, T') consists of two phylogenetic trees on the same set of taxa and that were constructed as follows. For each $t \in \{50, 100, 150, 200, 250, 300, 350\}$ we generated a random phylogenetic tree T on t taxa with *skew* $s \in \{50, 70, 90\}$, where the concept of skew is explained below. Then, for each $k \in \{5, 10, 15, 20, 25, 30, 35\}$ we applied k random TBR moves to T to obtain a phylogenetic tree T'. This ensures that $d_{\text{TBR}}(T, T') \leq k$. Note that equality might not hold, due to different random TBR moves potentially cancelling each other out. We produced 5 *replicates* of each tree pair: that is, for each parameter combination (t, s, k) we independently produced 5 different pairs of trees. This yields $7 \times 3 \times 7 \times 5 = 735$ pairs of trees in total. The skew s refers to the random generation of a binary phylogenetic tree via the following recursive process. (The process actually generates a rooted binary phylogenetic tree, but upon completion it is turned into an unrooted tree by suppressing the degree-2 root.) Starting with a rooted binary phylogenetic tree on two leaves, we place a taxon on one side of the root with probability $\frac{s}{100}$, and on the other side with probability $(1 - \frac{s}{100})$, and then recurse on the two sides of the root until the leaves are reached. Phylogenetic trees with a skew of 50 will be fairly balanced, corresponding to the standard Yule-Harding distribution (Harding 1971) while a skew of 90 will tend to produce a heavily skewed, more "linear" phylogenetic tree with smaller subtrees hanging off a main backbone. We include the skew parameter to incorporate more variety into the topology of the starting tree T.

For each tree pair (T, T'), we computed and collected the following core data:

D1 The number of taxa in the instance after application of the subtree reduction, henceforth denoted $s(T, T')$.

[2] See https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux, section 'Benchmarks'. Accessed 20th August 2021.

D2 The number of taxa in the instance after application of the subtree and chain reduction, henceforth denoted $sc(T, T')$.
D3 The number of taxa in the instance after application of all seven reductions, henceforth denoted $scn(T, T')$. For brevity we often call such tree pairs *fully reduced*.
D4 The number of parameter reductions that took place.
D5 The lower bound $d^{\ell}_{\mathrm{MP}}(T, T')$ on $d_{\mathrm{TBR}}(T, T')$ obtained by sampling convex characters for 10 s *after* the subtree reduction, since d_{MP} is also preserved by this reduction rule (Kelk et al. 2016). We computed this because at the outset of the experiments it was not clear whether we would be able to compute $d_{\mathrm{TBR}}(T, T')$ exactly. However, the exact TBR distance or a *lower* bound on this distance is needed to compute empirical kernel sizes (details follow below).
For each tree pair (T, T'), the exact value of $d_{\mathrm{TBR}}(T, T')$ if known. The distance was declared *known* if at least one of the following solution approaches terminated.

(i) Run uSPR for 5 min on the original tree pair (T, T').
(ii) Run uSPR for 5 min on the fully reduced tree pair obtained from (T, T') (and take into account the effect of any parameter reduction achieved during the kernelization).
(iii) Run Tubro for 5 min on the fully reduced tree pair obtained from (T, T') (and take into account the effect of any parameter reduction achieved during the kernelization). We only ran Tubro on those instances for which (i) and (ii) both failed to terminate, since it is not the goal of this article to directly compare the solving power of uSPR and Tubro.

Note that, if uSPR (or Tubro) produces an intermediate lower bound of k for a tree pair (T, T'), where k is the number of TBR moves that were applied to create the pair, it follows that $d_{\mathrm{TBR}}(T, T') = k$. However, if the solver does not also terminate in the allotted time, we do not consider such instances known. This is because the solvers do not have this upper bound information available outside the experimental framework that we describe here.

From D1–D6, we have computed various secondary statistics which are presented in more detail in Sect. 5. Most notably:

1. **The availability of exact TBR distances/difficult trees** The distribution of tree pairs whose TBR distance could not be computed exactly with uSPR or Tubro.
2. **Average percentage of remaining taxa** For each pair t and k, the average percentage of remaining taxa after the subtree reduction, after the subtree and chain reduction, and after all seven reductions over all 15 tree pairs with this parameter combination.
3. **Empirical kernel size** For each tree pair (T, T'), and for each of the different levels of reduction $s(T, T')$, $sc(T, T')$, and $scn(T, T')$, the number of taxa in the reduced instance divided by $d_{\mathrm{TBR}}(T, T')$ if the TBR distance is known and, otherwise, divided by $d^{\ell}_{\mathrm{MP}}(T, T')$.
4. **Parameter reductions** The distribution of tree pairs that have undergone at least one of the three reductions that trigger parameter reductions.
5. **The quality of the d_{MP} lower bound** The distribution of tree pairs (T, T') for which $d^{\ell}_{\mathrm{MP}}(T, T') < d_{\mathrm{TBR}}(T, T')$.

For a second dataset that comprises 90 tree pairs of larger size, we generated 5 replicates for each parameter combination (t, s, k) with $t \in \{500, 1000, 1500, 2000, 2500, 3000\}$, $s \in \{50, 70, 90\}$, and $k = 35$. To this end, we have followed the same approach as described for the main dataset and collected the same data and statistics for each tree pair. We will refer to this dataset as the *larger trees dataset*.

While we will focus on analyzing the main dataset in the following section, we use the larger trees dataset to confirm that our results do not only apply to pairs of phylogenetic trees with at most 350 taxa but instead describe general trends and observations that are not restricted to trees of a certain size.

We have made both our datasets, plus the spreadsheets describing our results, available on the page https://github.com/skelk2001/kernelizing-agreement-forests/. The GitHub page also includes a stand-alone implementation of the d_{MP} lower bound code, since we feel this is of independent interest. Source code for Tubro is available upon request.

5 Results and analysis

In each subsection below we combine the presentation of our results with some analysis and reflection.

5.1 The availability of exact TBR distances/difficult trees

We start by providing the number of tree pairs of both datasets for which we have computed the exact TBR distance, i.e. the number of tree pairs for which the TBR distance was declared *known*.

- 625 (85.0%) of the 735 tree pairs of the main dataset could be solved by uSPR in 5 min, when using the original (unreduced) trees.
- 646 (87.9%) of the 735 tree pairs could be solved by uSPR in 5 min, when using the fully reduced trees. In all cases where the original trees could be solved in 5 min, so too could the fully reduced trees.
- Of the remaining 89 tree pairs, Tubro could solve 51 after allowing the ILP solver Gurobi to run for 5 min. Hence, in total, the exact TBR distance was calculated for 646+51=697 (94.8%) of the tree pairs. For the remaining 38 tree pairs, d^{ℓ}_{MP} was used as a lower bound on the exact TBR distance. This lower bound was in particular used in the determination of empirical kernel sizes in Sect. 5.3.
- For the larger trees dataset, 86 tree pairs could be solved by running uSPR on the unreduced trees for 5 min and and additional 3 tree pairs could be solved by running uSPR on the reduced tree pairs. The remaining tree pair could not be solved by Tubro.

It is not the goal of this subsection to discuss how much kernelization does, or does not, help us to solve more challenging instances in practice. However, the experiments did yield some auxiliary insights in this direction, which we now describe. Applying all seven reductions allowed uSPR to solve 21 more instances of the main dataset than prior to reduction. While welcome, this is not a particularly large increase. This is not so surprising, because uSPR is a branching algorithm with exponential dependency on $d_{\mathrm{TBR}}(T, T')$ and only polynomial dependency on the number of taxa. Plus, uSPR contains an internal subtree reduction which already helps to reduce the number of taxa quite significantly. On the other hand, uSPR could potentially exploit parameter reduction, which does occur reasonably often (see Sect. 5.4). There is some one-sided evidence that parameter reduction did help. Specifically, 20 of the 21 instances that uSPR solved *after* reduction, exhibited parameter reduction. On the other hand, amongst the 89 instances that uSPR could still not solve after reduction, only 27.2% exhibited parameter reduction.

Regarding Tubro, we note that kernelization certainly helped in the following rather vacuous sense: the generation and solving time for the underlying ILP becomes prohibitively

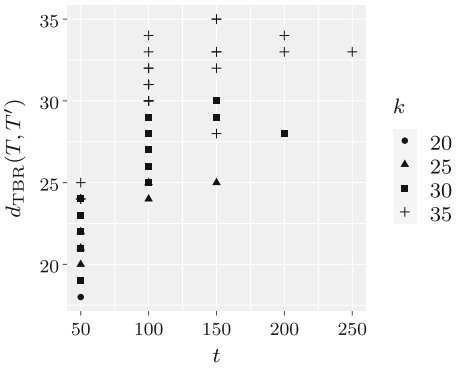


Fig. 4 Distribution of all 51 tree pairs of the main dataset that could be solved by Tubro but could not be solved by uSPR when applied to the fully reduced instances. Note that some of these 51 tree pairs have the same value for t, k, and d_{TBR} in which case their visualized data points coincide

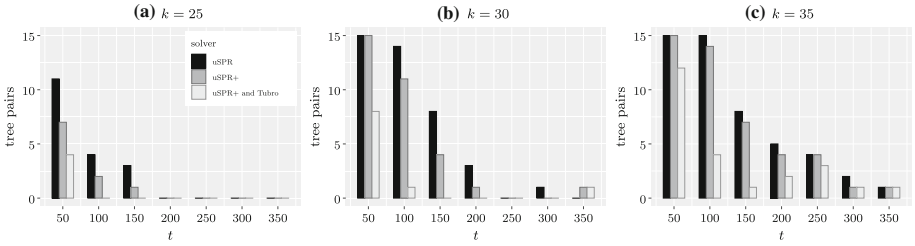


Fig. 5 Distribution of all 109 (black, uSPR), 88 (dark gray, uSPR+), and 38 (light gray, uSPR+ and Tubro) tree pairs of the main dataset with $k \geq 25$ that could not be solved exactly with uSPR applied to the subtree reduced tree pairs, with uSPR applied to the fully reduced tree pairs, and with Tubro applied to the fully reduced tree pairs, respectively. Note that one tree pair with $k = 20$ and $t = 50$ could not be solved with uSPR or uSPR+ but could be solved with Tubro

large for instances with more than (roughly) 70 taxa. Prior to kernelization, 85.7% of the tree pairs had more than 70 taxa, and after full kernelization only 50.5% had this property. As an unparameterized exponential-time algorithm, Tubro is naturally assisted more than an FPT branching algorithm by the reduction in instance size (i.e., $|X|$) achieved by kernelization. The 51 instances that Tubro could solve, but which uSPR could not, are summarized in Fig. 4. The average TBR distance of these 51 tree pairs is 27 with a standard deviation of 4.1. Of the 51 instances, 14 had 50 taxa, 22 had 100 taxa, 11 had 150 taxa, 3 had 250 taxa, and 1 had 250 taxa.

Three histograms that present the distribution of all tree pairs of the main dataset that could not be solved with the different solvers over all combinations of $k \in \{5, 10, 15, \ldots, 35\}$ and $t = \{50, 100, 150, \ldots, 350\}$ are shown in Fig. 5. The 110 tree pairs that could not be solved with uSPR applied to the subtree reduced tree pairs are presented by black bars. Similarly, the 89 (resp. 38) tree pairs that could not be solved with uSPR applied to the fully reduced tree pairs (resp. uSPR or Tubro applied to the fully reduced tree pairs) are presented by dark gray (resp. light gray) bars. Regardless of which solver was used, all unsolvable tree pairs were generated by applying $k \geq 25$ random TBR moves, except for one pair with $k = 20$ and $t = 50$ that could not be solved with uSPR but could be solved with Tubro (which is why the numbers 109 and 88 are reported in the figure caption). Turning to the 38 tree pairs that could not be solved by any of the solvers uSPR and Tubro, 12 had skew 50, 17 had skew 70 and 9 had skew 90, so tree pairs with a skew of 70 seem mildly over-represented. Furthermore, amongst the 38 unsolved instances, a majority of 24 tree pairs have the property $t = 50$. Given the parameterized running time of uSPR, which increases exponentially as a function

of k, it is not so surprising that uSPR had difficulties with the large TBR distance ($k \geq 25$) of the 38 unsolved tree pairs. It is less obvious why Tubro, which is based on ILP, finds these instances difficult. Clearly, the ILP generated by Tubro quickly becomes prohibitively large for tree pairs that have more than 70 taxa after kernelization. However, for $t = 50$ the ILP generated by Tubro will be comparatively small, and quick to generate, so this does not explain why Tubro struggles for some such tree pairs while it succeeds on others (14 of the 51 tree pairs that Tubro could solve but uSPR could not, had $t = 50$.) It is difficult to attach far-reaching conclusions to this, but in any case it seems that instances with a small number of leaves, where the TBR distance is high (as a function of the number of leaves) are potentially a challenge for both uSPR and Tubro. For uSPR, these instances might even be *harder* than instances with the same TBR distance, but more taxa. An *informal*, intuitive argument for this could be that, when the TBR distance is high and the number of taxa is low, the TBR moves required to turn one tree into the other heavily 'overlap' and 'interfere' with each other. Viewed through the lens of agreement forests, the underlying maximum agreement forest subsequently has very little structure, causing uSPR and Tubro to approach their worst-case behaviour. We return to these tractability issues in Sect. 6.

5.2 Average percentage of remaining taxa

To evaluate whether or not the five new reductions further reduce phylogenetic trees that have already been reduced by the subtree and chain reduction, we have analyzed the percentage of remaining taxa of all 735 tree pairs after (i) the subtree reduction, (ii) the subtree and chain reductions, and (iii) all seven reductions. More specifically, for each pair t and k with $t \in \{50, 100, 150, \ldots, 350\}$ and $k \in \{5, 10, 15, \ldots, 35\}$, we have calculated the three average percentages

$$\frac{100}{t} \cdot s(T, T') \qquad \frac{100}{t} \cdot sc(T, T') \qquad \frac{100}{t} \cdot scn(T, T')$$

over all 15 tree pairs (T, T') with parameter combination (t, s, k), where $s \in \{50, 70, 90\}$. The results are summarized in Fig. 6. For example, the seven reductions reduce tree pairs with $t = 350$ and $k = 5$ by about 90% and tree pairs with $t = 350$ and $k = 35$ by about 60%. Except for the two parameter combinations $(50, s, 30)$ and $(50, s, 35)$, Fig. 6 shows that applying all seven reductions always results (on average) in smaller tree pairs than applying the subtree and chain reductions only. Observe that, regardless of k, the percentage of the number of remaining taxa decreases as t increases. For example, for $k = 20$ and $t = 50$, we have $\frac{100}{50} \cdot scn(T, T') \approx 80\%$, and for $k = 20$ and $t = 350$, we have $\frac{100}{350} \cdot scn(T, T') \approx 27\%$. An analogous trend applies to all other values of k. This can be partially explained by recalling that, when applied to exhaustion, the seven reduction rules guarantee that the resulting reduced instance has no more than $11d_{\mathrm{TBR}} - 9$ taxa (Kelk and Linz 2020). So, taking k as a proxy for d_{TBR}, the ratio $(11k - 9)/t$ decreases as t increases, if k is fixed. Hence, the curve behaves like $1/t$ (a hyperbola) for fixed k; decreasing, but at an ever slower rate. A similar observation holds if we only apply the subtree and chain reduction, since the reduced instances are then guaranteed to have at most $15d_{\mathrm{TBR}} - 9$ taxa. Interestingly, when only the subtree reduction is applied, the solid curves in Fig. 6 have a very similar downward-sloping tendency as the sc and scn curves, despite the fact that, when applied in isolation there is no $g(k)$ theoretical upper

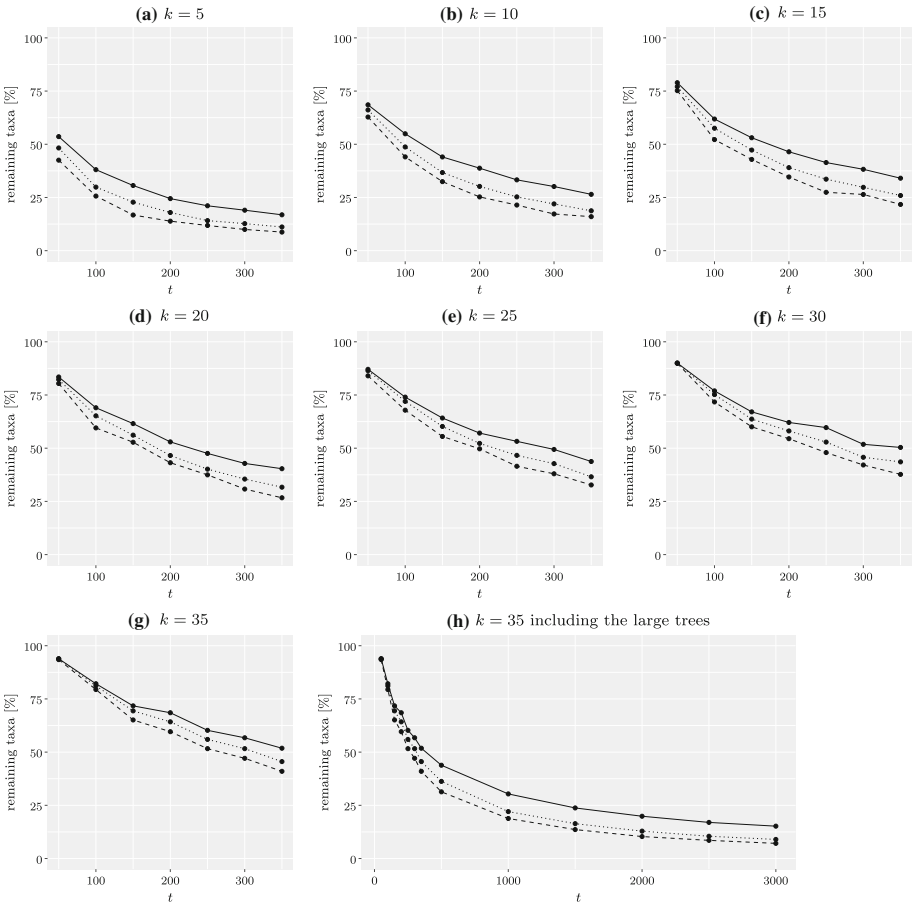


Fig. 6 The average percentage of taxa that remains after the subtree reduction (solid line), after the subtree and chain reduction (dotted line), and after all seven reductions (dashed line) depending on t

bound on the size of the reduced instance[3]. To gain more insight into this phenomena, we have redone the analysis underlying Fig. 6d, but separately for each skew value $s \in \{50, 70, 90\}$.

The results, presented in Fig. 7, suggest that for $s \in \{50, 70\}$ similar performance is observed, but that at $s = 90$ the achieved reduction is decreasing when only the subtree reduction is applied. It seems that high skew militates against the creation of common subtrees. Intuitively, one would expect the chain reduction to work well at $s = 90$, since high skew would seem to support the creation of multiple common long chains; perhaps the two effects (i.e. less effective subtree reduction, possibly more effective chain reduction) cancel each other out.

Returning to Fig. 6, a further observation is as follows. Recalling that $g(k)$ denotes the theoretical upper bound on the size of the kernel, the fact that $g(k)/t$ decreases for fixed k

[3] Such a bound cannot exist. Consider, for example, the situation when T and T' are caterpillar trees—every internal vertex is incident to at least one taxon—on $n + 2$ taxa, $n \geq 3$, with taxa ordered $x, 1, 2, \ldots, n, y$ and $y, 1, 2, \ldots, n, x$ in T and T', respectively. Irrespective of n, the TBR distance between T and T' is two, the subtree reduction does nothing here, and the chain reduction collapses both trees to 5 taxa.

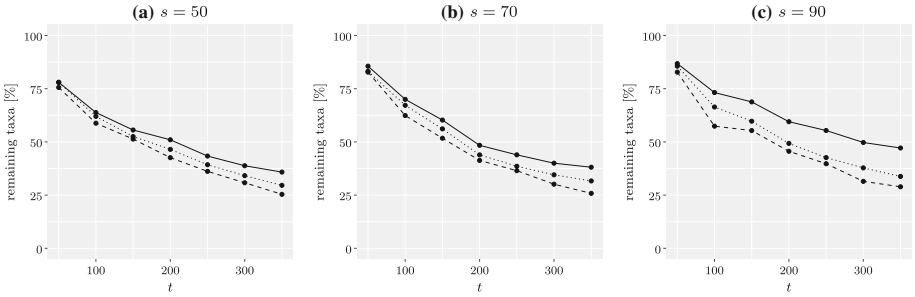


Fig. 7 The average percentage of taxa that remains after the subtree reduction (solid line), after the subtree and chain reduction (dotted line), and after all seven reductions (dashed line) for all tree pairs with $k = 20$, for different skew values

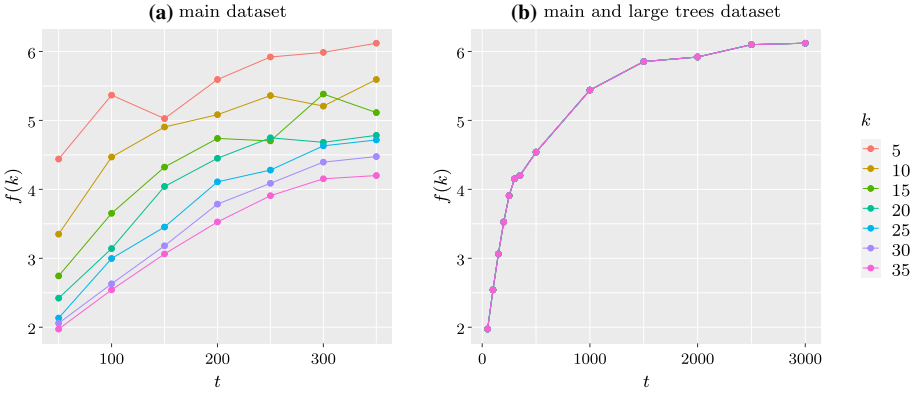


Fig. 8 Empirical estimates $f(k)$ of kernel size, depending on t, for the fully reduced tree pairs

and increasing t explains the downward-sloping shape of the sc and scn curves, but it only has limited explanatory power. If t is smaller than $11d_{\text{TBR}} - 9$, then the reduction rules offer no guarantees at all. For example, for $k = 35$, we have $11k - 9 = 376$, which is larger than any t used in our main dataset. Nevertheless, as shown in Fig. 6g, significant reduction is still achieved. This suggests that the upper bound on kernel size may, in practice, be much smaller than $11d_{\text{TBR}} - 9$, a point we elaborate on in Sect. 5.3.

Finally, a brief reflection on the fact that the (50, s, 30) and (50, s, 35) instances exhibited little reduction and that the difference between sc and scn was negligible. Such instances have an extremely high TBR distance relative to the number of taxa. This seems to destroy any of the structures that are targeted by the reduction rules. Interestingly, such instances are not only difficult to reduce, they also appear potentially difficult to solve, as we saw in the earlier section. In both cases this seems linked to the phenomenon that, for such instances, a maximum agreement forest is likely to have many very small components. This is a point we will return to later.

5.3 Empirical kernel size

By Kelk and Linz (2020, Theorem 6), we have a theoretical upper bound of $g(k) = 11d_{\text{TBR}}(T, T') - 9$ on the size of the TBR kernel if T and T' are fully reduced. Hence,

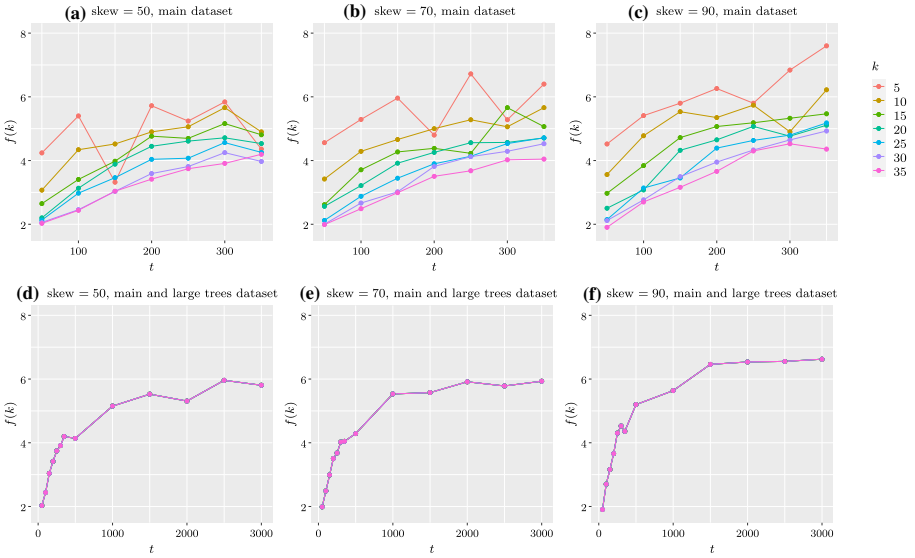


Fig. 9 Empirical estimates $f(k)$ of kernel size, depending on t, for the fully reduced tree pairs broken down by different skew values

$scn(T, T') \leq 11d_{\mathrm{TBR}} - 9 \leq 11k - 9$. To evaluate how well this theoretical kernel compares to the empirical kernel size obtained from our experimental analysis, we have computed

$$f(k) = \frac{scn(T, T')}{\theta(T, T')}$$

for all tree pairs of the main and larger trees dataset, where $\theta = d_{\mathrm{TBR}}$ if $d_{\mathrm{TBR}}(T, T')$ is known and, otherwise, $\theta = d^{\ell}_{\mathrm{MP}}$. The results are summarized in Figs. 8 and 9 . For each pair t and k with $t \in \{50, 100, 150, \ldots, 350\}$ and $k \in \{5, 10, 15, \ldots, 35\}$, we have plotted the average value of $f(k)$ over all 15 tree pairs with parameter combinations (t, s, k), where $s \in \{50, 70, 90\}$ in Fig. 8a. Across all 735 tree pairs of the main dataset, the maximum empirical kernel size is 9.8. Similarly for $k = 35$ and each

$$t \in \{50, 100, 150, \ldots, 350, 500, 1000, 1500, \ldots, 3000\},$$

we have plotted the average value of $f(k)$ over all 15 tree pairs with parameter combinations (t, s, k) in Fig. 8b. Across all 195 tree pairs (i.e. all 90 tree pairs of the large trees dataset and all 105 tree pairs of the main dataset with $k = 35$), the maximum empirical kernel size is 7.37. In summary, our results show that the empirical kernel size $f(k)$ is, regardless of the parameter combination (t, s, k), always significantly smaller than the theoretical upper bound. Note that the $g(k)$ bound in Kelk and Linz (2020) is tight, in the sense that reduced instances can be constructed with *exactly* $11d_{\mathrm{TBR}} - 9$ taxa. This implies that the observed gap between the empirical and theoretical bound cannot be closed by strengthening the analysis in Kelk and Linz (2020). While $f(k)$ increases with t, the slope of the curve drops off for very large t as shown in Fig. 8b. However, even for $t = 3000$, $f(k) = 6.12$ is still much smaller than the theoretical upper bound of 11. If we compare the empirical kernel sizes for a fixed t and different values of k, we see in Fig. 8a that $f(k)$ decreases with increasing k. One possible explanation for this is as follows. For a fixed t, the number of tree pairs in the dataset that have the property $t/k < c$ (for a given constant c) clearly increases as k increases.

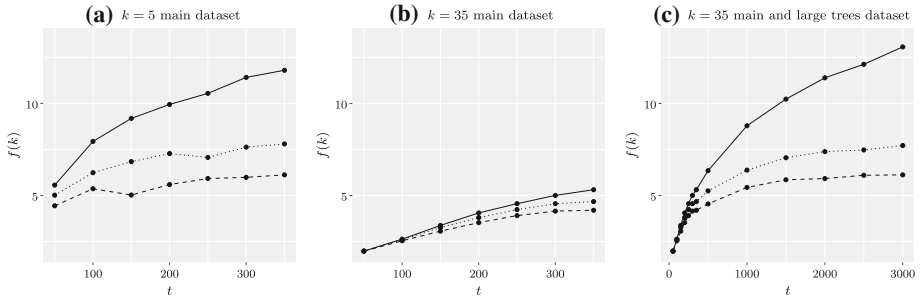


Fig. 10 Empirical estimates $f(k)$ of kernel size, depending on t, for the subtree reduced (solid), the subtree and chain reduced (dotted), and the fully reduced (dashed) tree pairs

For example, for $k = 35$ all the trees with $t \in \{50, 100\}$ satisfy the inequality if we take $c = 3$; but at $k = 10$ and $c = 3$ none of the trees in the dataset do, because all the trees in the dataset have at least 50 taxa. Now, if we take k as an estimator of d_{TBR}, we see that $f(k) = \frac{scn(T,T')}{\theta(T,T')}$ will *a priori* be at most c for tree pairs satisfying the inequality, because $scn(T, T') \leq t$ and $\theta(T, T') \approx k$. Possibly this has the effect of pulling the $f(k)$ curves downwards for increasing k i.e. because a higher proportion of the trees in the dataset have a small number of taxa relative to d_{TBR}, and thus contribute very low $f(k)$ values.

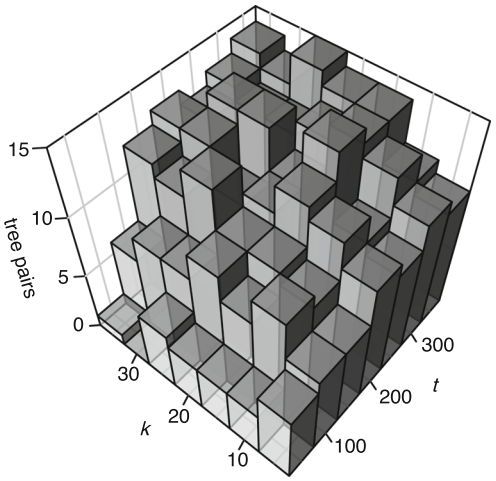
To evaluate the impact of different skew values on the $f(k)$ values, we have repeated the analysis that underlies Fig. 8 for each of the three skew values {50, 70, 90}. The results are shown in Fig. 9 with curves that are overall very similar to those shown in Fig. 8. Moreover, for fixed values of k and t, we see that $f(k)$ increases very slowly as s increases which may again be the result of the subtree reduction being (on average) less effective when applied to trees pairs that are heavily skewed (possibly to the extent of cancelling out enhanced performance of the chain reduction—but whether this indeed occurs requires further investigation).

Finally, we note that similar to the analysis that underlies Fig. 8, empirical kernel sizes can also be computed for the tree pairs that have been reduced under the subtree reduction only, or under the subtree and chain reduction only. For a selection of tree pairs of both data sets, these are shown in Fig. 10. In particular, we note that the *sc* curve remains well below the worst-case bound of 15; in Fig. 10c the curve seems to be flattening at around $f(k) = 7$.

These results hint at the possibility that the empirical kernel size is determined by more parameters than just d_{TBR}. It is difficult to say what these hidden parameters might be. Popular 'local' phylogenetic parameters such as *level* (Bordewich et al. 2017) do not explain the phenomenon, since the seven reduction rules are largely uninfluenced by low level. The phenomenon is probably linked to the fact that the tight instances described in Kelk and Linz (2019, 2020) were extremely carefully engineered. It is very unlikely that such instances will occur in our experimental setting, so the worst-case kernel size is unlikely to occur. To understand this phenomenon further it will be necessary to carefully analyze such tight instances and formalize how (and, quantitatively, how far) they differ from trees generated in our experimental setting; we defer this to future work.

5.4 Parameter reductions

We have analyzed the frequency with which the five new reductions have triggered parameter reductions among all 735 tree pairs and observed that 499 tree pairs were reduced by at least one application of Reduction 3, 4, or 5. The maximum number of parameter reductions for a



parameter red.	tree pairs
0	236
1	217
2	133
3	89
4	34
5	17
6	6
7	3
skew	tree pairs
50	169
70	163
90	167

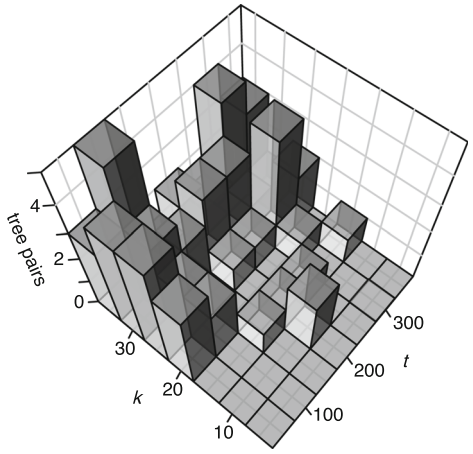
Fig. 11 Left: Distribution of all 499 tree pairs of the main dataset that were reduced by at least one application of Reduction 3, 4, or 5. Right: Summary of all tree pairs that were reduced by at least one application of Reduction 3, 4, or 5 depending on the total number of such reductions and the skew of the tree pairs over all 735 tree pairs

single tree pair is seven. A histogram that presents the distribution of the 499 tree pairs over all combinations of $k \in \{5, 10, 15, \ldots, 35\}$ and $t = \{50, 100, 150, \ldots, 350\}$ as well as a table that summarizes the frequency of parameter reductions over all 735 tree pairs is shown in Fig. 11. Each of the 3 tree pairs that have undergone seven parameter reductions have a parameter combination with $t \geq 250$ and $k \geq 30$. Figure 11 indicates that the number of tree pairs that have triggered at least one parameter reduction increases as both t and k grow. To confirm this trend, we have also analyzed the amount of parameter reductions for all tree pairs of the larger trees dataset. For this dataset, all 90 tree pairs have been reduced by at least one application of Reduction 3, 4, or 5 and the maximum number of such reductions applied to a single tree pair is ten. The associated tree pair has the parameter combination $t = 3000$ and $k = 35$. These results show that not only the ordinary new reductions that preserve the TBR distance (these are Reductions 6 and 7) enhance the power of the subtree and chain reduction but that the same holds for the other three reductions.

It is known that two of the parameter-reducing reductions, Reductions 4 and 5, trigger an immediate subsequent application of the subtree rule, but beyond this it is too difficult to argue analytically why any of the reduction rules should trigger each other. This is because the seven reduction rules target rather different local structures, and after applying a reduction rule the local structure is either destroyed or remains incompatible with other reduction rules. The phenomenon of parameter reduction occurring repeatedly for a single pair of trees is therefore quite possibly caused by multiple parts of the trees triggering reduction rules independently. Understanding whether this is indeed what happens, or whether prioritizing certain reduction rules over others works well in practice, requires further empirical research.

5.5 The quality of the d_{MP} lower bound

To evaluate the quality of d^{ℓ}_{MP}, which we have used as a lower bound on the TBR distance throughout our experimental study, we have compared $d^{\ell}_{\mathrm{MP}}(T, T')$ and $d_{\mathrm{TBR}}(T, T')$



$d_{\mathrm{TBR}}(T,T') - d^{\ell}_{\mathrm{MP}}(T,T')$	tree pairs
0	635
1	57
2	4
3	1
skew	tree pairs
50	23
70	19
90	20

Fig. 12 Left: Distribution of all 62 tree pairs (T, T') with $d^{\ell}_{\mathrm{MP}}(T, T') < d_{\mathrm{TBR}}(T, T')$. Right: Summary of the difference between d_{TBR} and d^{ℓ}_{MP} over all 697 tree pairs with known TBR distance, and summary of all 62 tree pairs whose d_{MP} lower bound is less than their TBR distance depending on the skew of these tree pairs

for all 697 tree pairs (T, T') for which we could calculate the TBR distance exactly (see Sect. 5.1). Only 62 (that is 8.9%) out of 697 tree pairs of the main dataset have $d^{\ell}_{\mathrm{MP}}(T, T') < d_{\mathrm{TBR}}(T, T')$. Among these 62 pairs, a majority of 92% have $d_{\mathrm{TBR}}(T, T') - d^{\ell}_{\mathrm{MP}}(T, T') = 1$. The maximum difference between $d_{\mathrm{TBR}}(T, T')$ and $d^{\ell}_{\mathrm{MP}}(T, T')$ is 3 over all 62 tree pairs. A histogram that presents the distribution of these 62 tree pairs over all parameter combinations of $k \in \{5, 10, 15, \ldots, 35\}$ and $t = \{50, 100, 150, \ldots, 350\}$ is shown in Fig. 12. The tree pair with $d_{\mathrm{TBR}}(T, T') - d^{\ell}_{\mathrm{MP}}(T, T') = 3$ has the parameter combinations $t = 350$ and $k = 35$. In comparison, 48 out of all 90 tree pairs of the large trees dataset have $d^{\ell}_{\mathrm{MP}}(T, T') < d_{\mathrm{TBR}}(T, T')$. However, the maximum difference between $d_{\mathrm{TBR}}(T, T')$ and $d^{\ell}_{\mathrm{MP}}(T, T')$ remains 3. These results verify that d^{ℓ}_{MP} is an effective lower for computing d_{TBR} and that the difference between $d_{\mathrm{TBR}}(T, T')$ and $d^{\ell}_{\mathrm{MP}}(T, T')$ grows very slowly, even for large trees with up to 3000 leaves.

Another point worth drawing attention to, is the *efficiency* of computing d^{ℓ}_{MP}. In our main dataset d^{ℓ}_{MP} was, for 679 tree pairs, after 10 s greater than or equal to the best lower bound that uSPR had computed after 5 min of iterative deepening.

6 Discussion: shattered forests?

Recall that there were 38 pairs of trees for which neither uSPR nor Tubro could compute d_{TBR} in 5 min. These all had high d_{TBR} ($k \geq 25$). One of the most striking tractability insights from the experiments is that, amongst these instances, 24 had only 50 taxa. Indeed, Fig. 5 suggests that, within the group of 38 instances, and for fixed k, trees with fewer taxa are *more* likely to confound the solvers than trees with more taxa. This is noteworthy and, as explained in Sect. 5.1, it is not obvious why uSPR or Tubro would exhibit this behavior. Less surprisingly, instances with $t = 50$ and high TBR distance ($k \in \{30, 35\}$) are largely unaffected by the seven reduction rules (see Sect. 5.2), while all other parameter combinations exhibit higher levels of reduction. The emerging picture is that tree pairs with a small number of taxa t

but (very) high TBR distance relative to t, are both hard to solve, and hard to reduce. The common factor here, we suspect, is that for such pairs of trees a maximum agreement forest will necessarily have many components, of which many will be small. Manual inspection of such instances suggests that many components of the forest indeed contain only a single, or perhaps two, taxa. Such "shattered" forests will only very occasionally trigger the seven reduction rules. Apparently, such forests and their lack of topological structure can also cause both the combinatorial branching strategy of uSPR and the ILP-based branching of Tubro severe problems. FPT algorithms such as uSPR are not designed to run quickly when the parameter in question (here, d_{TBR}) is high, and the focus when developing such algorithms is usually to ensure fast running time when the size of the instance is *large* relative to the parameter. Here we have a high parameter combined with *small* instances. This requires further research and different algorithmic techniques.

It is probably relevant that such instances increasingly start to resemble *random* pairs of trees; the literature on the expected number of components in maximum agreement forests of random pairs of trees is therefore worth exploring (Atkins and McDiarmid 2019).

7 Conclusions and future work

In this article we have demonstrated that reduction rules for TBR distance have the potential to significantly reduce the size of instances, and that theoretically stronger reduction rules do have clear added value in practice. Our experimental results also highlight that the empirical bound on the size of the TBR kernel is significantly lower than that predicted in the worst-case ($15k - 9$ and $11k - 9$ respectively, for the two different sets of reduction rules). Lastly, we have shown that parameter reduction occurs quite often, and that a sampling strategy based on sampling convex characters quickly yields strong lower bounds on d_{TBR}.

In addition to the phenomenon of "shattered forests" raised in the previous section, a number of interesting questions have emerged from our work concerning (i) kernelization and (ii) the actual computation of TBR distance. Regarding kernelization, it is natural to ask why the empirical kernel bound is much better than the worst-case bound. Clearly, the carefully-constructed tight instances described in Kelk and Linz (2019, 2020) are unlikely to occur in an experimental setting such as ours. Such phenomena are pervasive in theoretical computer science. Nevertheless, can we rigorously explain *why* the experimentally-generated instances can be more successfully reduced? One way to tackle this would be to search for additional parameters which, when combined with d_{TBR}, produce a more accurate prediction of the obtained kernel size. We currently do not have any concrete candidates for what these parameters could be, but understanding how the tight instances from Kelk and Linz (2019, 2020) differ from those in our experimental setting, and quantifying this dissimilarity, is likely to yield insights in this direction. Next, it is natural to ask: is it possible, by developing new rules, to obtain a kernel smaller than $11k - 9$, and does it make sense to implement these rules in practice? Breaking the $11k - 9$ barrier is likely to be primarily a theoretical exercise, guided by the limits of the counting arguments in Kelk and Linz (2020). However, perhaps an empirical examination of how solvers such as uSPR tackle fully reduced instances could offer some extra clues, at least on the question of whether any future reduction rules will be effective in practice.

On the computational side, there is still room for improvement. While the combination of uSPR and Tubro is capable of solving most of the trees in our experimental dataset, there still exist pairs of sometimes small trees where neither uSPR or Tubro can compute d_{TBR} in 5 min.

Perhaps the branching factor in the FPT branching algorithm underpinning uSPR (Chen et al. 2015; Whidden and Matsen 2018) can be lowered by a deeper study of the combinatorics of agreement forests. It also seems important to understand why certain pairs of trees trigger multiple parameter reductions. On the engineering side we are optimistic that more advanced ILP engineering techniques can be used to speed up Tubro. Future research should focus on continuing the strategy of blending and merging existing techniques into an ensemble that we adopted in this article. These techniques include kernelization to reduce the size of instances (and to generate useful combinatorial insights, such as chain preservation), fast generation of lower and upper bounds, FPT branching algorithms, ILP, and other exponential-time algorithms. In this article the techniques were coupled fairly loosely. We anticipate that an ensemble in which the techniques are more deeply integrated, i.e. interleaved, will yield further speedups. This is a challenging engineering task, but also a challenging mathematical one, since it is not always easy to translate intermediate solutions or bounds produced by one of the techniques to another. For example, as soon as a branching algorithm starts cutting edges in the trees, the instance becomes a more general type of instance: a pair of forests, rather than a pair of trees. To adapt the reduction rules from Kelk and Linz (2020) to forests will require a significant theoretical effort.

Acknowledgements We would like to thank the participants of the *Algorithms and Complexity in Phylogenetics* seminar in 2019 (Dagstuhl Seminar 19443) and Schloss Dagstuhl for hosting the seminar. We also thank A. Alhazmi for useful conversations on the topic of this paper and an anonymous referee for their helpful comments.

A Tubro: Using kernelization and Integer Linear Programming to compute $d_{\mathbf{TBR}}$

As is well known, kernelization does not in itself solve a problem, it simply reduces it in size. To solve the kernel an (efficient) exponential-time algorithm is required. Throughout the experimental section we mainly used uSPR to compute TBR distances, and this was primarily to be able to compute $f(k)$ values (i.e. estimates of the empirical kernel size). As part of our work, we experimented not just with kernelization but also with an alternative exact algorithm for computing the TBR distance exactly. The result is our package Tubro, which incorporates all seven reductions and can, if desired, output the reduced trees to be solved by another package. However, it also incorporates its own exact algorithm, based on augmenting a core Integer Linear Programming (ILP) formulation with certain additional features. Before we present the ILP formulation, and describe the various augmentations as well as Tubro's strengths and weaknesses, we need a couple new definitions.

Let T be a phylogenetic tree on X. A *quartet* is a phylogenetic tree with exactly four leaves. For example, if $\{a, b, c, d\} \subseteq X$, we say that $ab|cd$ is a quartet of T if the path from a to b does not intersect the path from c to d in T. Note that, if $ab|cd$ is not a quartet of T,

then either $ac|bd$ or $ad|bc$ is a quartet of T. If $ab|cd$ is a quartet of T, we say that T *displays* $ab|cd$. As a consequence, T displays exactly $\binom{n}{4}$ quartets.

A.1 A polynomial-size Integer Linear Programming formulation for MAF

The ILP formulation we use is adapted from Wu (2009). Both our formulation and the formulation in Wu (2009) are based on the idea of cutting a minimum number of edges in one of the trees, such that an agreement forest is obtained. To this end, decision variables represent which edges are cut. The formulation in Wu (2009) works on *rooted* binary phylogenetic trees, leading to two different types of constraints. Here we are working with *unrooted* binary phylogenetic trees. As we shall see this allows us to obtain a simplified ILP in which there is only one type of constraint. We start by describing the formulation and proving its correctness.

Let T and T' be the two phylogenetic trees on X with $n = |X|$. We select arbitrarily one of the trees; here we take T. Let $E(T)$ be the set of edges in T. For two taxa $x, y \in X$, let $P_T(xy)$ be the edges on the unique path from x to y in T. Furthermore, let Q be the set consisting of those quartets which are displayed by T but not by T'. For each $e \in E(T)$ the ILP contains a binary decision variable x_e, so $2n - 3$ such variables in total. There is one constraint for each quartet in Q. The number of constraints thus depends on how different T and T' are, but is in any case $O(n^4)$. The ILP formulation has a Hitting Set flavor:

$$\begin{aligned} &\text{minimize} && \sum_{e \in E(T)} x_e && \\ &\text{subject to} && \sum_{e \in P_T(ab) \cup P_T(cd)} x_e \geq 1 && \text{for each } ab|cd \in Q \\ &\text{and} && x_e \in \{0, 1\} && \text{for each } e \in E(T) \end{aligned}$$

Rather than proving (only) that the optima of the ILP coincides with $d_{\text{MAF}}(T, T')$, we prove the following slightly more general statement, which allows also non-optimal solutions to the ILP to be mapped to agreement forests. This is crucial to extract valid agreement forests even if it takes too long for the ILP solver to reach optimality.

Theorem 2 *Let T and T' be two phylogenetic trees on X. An agreement forest F for T and T' with $|F| = t$ induces a feasible solution to the ILP with objective function value $t - 1$. Furthermore, a feasible solution to the ILP with objective function value t' induces an agreement forest F of T and T' with $|F| \leq t' + 1$ components.*

Proof Let F be an agreement forest of T and T' containing t components. By the definition of an agreement forest, there exists a subset E_F of $E(T)$ with $|E_F| = t - 1$ such that deleting exactly the edges of E_F in T results in a graph that contains t connected components and the partition of X induced by the taxa in these components is exactly equal to F. Note that E_F is not necessarily unique. We argue that setting the decision variables corresponding to E_F to 1, and all other decision variables to 0, yields a feasible solution to the ILP. Towards a contradiction, assume that this is not the case. Then there is a quartet $ab|cd$ such that T displays $ab|cd$, T' does not display $ab|cd$, and none of the decision variables corresponding to edges in $P(ab) \cup P(cd)$ are set to 1. Observe that, for each component $B \in F$, $\{a, b, c, d\} \nsubseteq B$, because otherwise $T|B \neq T'|B$ contradicting that F is an agreement forest for T and T'. So $\{a, b, c, d\}$ intersects at least two components of F. If $\{a, b\} \subseteq B$ and $\{c, d\} \subseteq B'$, where $B \neq B'$, then because $T[B]$ and $T[B']$ (resp. $T'[B]$ and $T'[B']$) are vertex-disjoint in T (resp. T') both T and T' display $ab|cd$; again a contradiction. In fact, the only two possibilities are (1) that each taxon in $\{a, b, c, d\}$ intersects a different component of F, and

(2) three of $\{a, b, c, d\}$ occur in a component B and the remaining taxon in B', where $B' \neq B$. Hence, regardless which of (1) and (2) applies, a and b occur in different components of F and/or c and d occur in different components of F. Suppose without loss of generality that a and b occur in different components of F. Then $P(ab) \cap E_F \neq \emptyset$ which implies that there exists an edge in $P(ab)$ whose corresponding decision variable is set to 1; yielding a final contradiction.

For the second statement, assume that we have a feasible solution to the ILP with objective function value t'. Let E_I be the edges of T corresponding to decision variables that have been set to 1 in this solution. Let P be the partition of X induced by the connected components of $E(T)\backslash E_I$ after deleting any connected components that do not contain any taxa, if they exist[4]. Clearly, $|P| \leq t' + 1$, since the deletion of an edge increases the number of connected components by at most one. We claim that P is in fact an agreement forest of T and T'. Once again towards a contradiction, assume this is not the case. Suppose that condition (1) in the definition of an agreement forest is violated. Then, there exists a block $B \in P$ such that $|B| \geq 4$ and $T|B \neq T'|B$. It follows that there exist 4 taxa $\{a, b, c, d\} \subseteq B$ such that, without loss of generality, $ab|cd$ is displayed by $T|B$ (and hence by T) but not by $T'|B$ (and hence not by T'). But none of the edges on $T[B]$ are in E_I. In particular, none of the edges on the path from a to b and none of the edges on the path from c to d in T have been cut, contradicting the feasibility of the ILP solution. Now, suppose that condition (2) is violated. Recall that, in T, the images of B and B' do not intersect, by construction. Thus there exist two distinct blocks $B, B' \in P$ such that the two embeddings $T'[B]$ and $T'[B']$ are not vertex-disjoint in T'. In fact, they are not edge-disjoint, due to the fact that T' is binary. In turn, this implies that $|B|, |B'| \geq 2$. Let $e = \{u, v\}$ be any edge in T' that is shared by $T'[B]$ and $T'[B']$. Deleting e naturally induces a partition of B into B_u and B_v, where B_u (resp. B_v) are those taxa in B that are closer in T' to u than v (resp. closer in T' to v than u). Observe that each of B_u and B_v contains at least one taxon. Let B'_u and B'_v be defined analogously with respect to B'. Furthermore, let $a \in B_u$, $b \in B_v$, $c \in B'_u$ and $d \in B'_v$. Since $T[B]$ and $T[B']$ do not intersect in T, and $\{a, b\} \subseteq B$, and $\{c, d\} \subseteq B'$, it follows that T displays $ab|cd$. However, T' displays $ac|bd$, because there is a path from a to c passing through u and a path from b to d passing through v and these two paths are disjoint. Hence, the ILP included a constraint to cut at least one edge on the paths in T from a to b and from c to d. But no such edge was cut, because $\{a, b\} \subseteq B$ and $\{c, d\} \subseteq B'$; a contradiction. □

The next corollary is an immediate consequence of Theorem 2.

Corollary 1 *The optimum value computed by the ILP equals* $d_{\mathrm{TBR}}(T, T') = d_{\mathrm{MAF}}(T, T') - 1$.

Note that, if it is known a priori that there exists a maximum agreement forest in which a certain edge e of T is *not* cut, this can easily be enforced by adding the constraint $x_e = 0$ (or better by removing x_e from the set of decision variables). As we shall see in due course, Tubro can make good use of this, for example to stipulate that it is unnecessary to cut edges in preserved common chains even if the chains themselves cannot be further reduced. More complex restrictions on feasible solutions, such as "at least one of edge e and e' must be cut", can easily be added using standard ILP modeling techniques.

[4] An optimal solution to the ILP never induces such taxa-free connected components, but a sub-optimal solution might.

A.2 High-level description of Tubro

Tubro is highly configurable, with many options that can be switched on or off. The core functionality can be summarized as follows.

1. First, the instance is kernelized by applying all seven reductions.
2. Next, the ILP formulation described in the previous section is generated.
3. The ILP formulation is simplified using *chain preservation* (see below).
4. A simple greedy algorithm is applied to produce an agreement forest that is not-necessarily maximum, which gives an upper bound on d_{TBR}. This is the classical greedy algorithm for Hitting Set instances (Chvatal 1979): select a decision variable not yet set to 1 which intersects with a maximum number of unsatisfied constraints (cq. resolves a maximum number of as yet unresolved quartet conflicts which are quartets displayed by only one of T and T').
5. The ILP is fed to the ILP solver (Gurobi 2020). In the experiments we used Gurobi version 8.1.1. and warm-started it with the upper bound from the greedy algorithm described in Step 4.
6. While Gurobi is solving, the d_{MP} lower bound sampling strategy algorithm is run in parallel and every time an improved lower bound is found, this is communicated to Gurobi.

The chain preservation mentioned above in Step 3 leverages (Kelk and Linz 2020, Theorem 5), which is as follows:

Theorem 3 *Let T and T' be two phylogenetic trees on X. Let K be an (arbitrary) set of mutually taxa-disjoint chains that are common to T and T'. Then there exists a maximum agreement forest F of T and T' such that*

1. *every n-chain in K with $n \geq 3$ is preserved in F, and*
2. *every 2-chain in K that is pendant in at least one of T and T' is preserved in F.*

In Kelk and Linz (2020) the theorem was used as part of a more comprehensive argument to prove the correctness of several of the new reduction rules introduced there. Interestingly, even after all reduction rules have been applied to exhaustion, there can exist common chains that, although they cannot be reduced further in length, still obey the above theorem. As a result, it is safe to assume the existence of a maximum agreement forest in which no such chain is split across two or more components of this forest. Translated into the language of ILP, this means that for each edge e contained in such a chain, we can *a priori* set $x_e = 0$ in the ILP. These variables can thus be removed from the ILP, simplifying the system. Tubro uses a simple greedy strategy to select a set of mutually taxa-disjoint chains with maximum total length, thus optimizing the number of variables that can be removed. In our analysis of the main dataset, chain preservation was switched *on*.

Although a full explanation is beyond the scope of this paper, we observed that chain preservation does remove quite a lot of variables, *even in fully reduced instances*. To illustrate this, we performed a secondary experiment on all fully-reduced instances from the main dataset where the original trees had 50, 100, or 150 taxa. This was in total 315 tree pairs. We observed that on average the number of variables in the ILP was reduced by 26.29% (standard deviation of 8.77) when chain preservation was switched on, compared to when it was switched off. This suggests that a significant number of common chains survive, that cannot be further reduced by the new reduction rules, but which *do* fall under the chain preservation theorem. Indeed, the counting argument used in Kelk and Linz (2020) to obtain the $11k-9$ upper bound on kernel size is based on the possibility of $O(k)$ irreducible 3-chains

or 2-chains existing. These chains cannot be reduced by the existing reduction rules, but (for those that fall under the chain preservation theorem for a given set of mutually taxa-disjoint chains) we can still algorithmically exploit the fact that they are preserved in some maximum agreement forest, by stipulating that the edges in the chains should not be cut.

A.3 Optional extra: cluster reduction

Let T and T' be two phylogenetic trees on X. For $Y \subset X$ with $|Y| \geq 2$ and $|X - Y| \geq 2$, we say that Y is a *cluster* of T if there exists a single edge in T whose deletion disconnects T into two parts such that the leaves of one part are bijectively labeled by elements in Y while the leaves of the other part are bijectively labeled by elements in $X - Y$. Now, let Y be a cluster of both T and T'; this is often referred to as a *common cluster*. In Bordewich et al. (2017) a divide-and-conquer reduction rule is presented which, at a high level, computes $d_{\text{MAF}}(T, T')$ by computing $d_{\text{MAF}}(T|Y, T'|Y)$ and $d_{\text{MAF}}(T|X - Y, T'|X - Y)$ separately. It can be shown that $d_{\text{MAF}}(T, T')$ is either equal to,

$$d_{\text{MAF}}(T|Y, T'|Y) + d_{\text{MAF}}(T|X - Y, T'|X - Y) \tag{1}$$

or

$$d_{\text{MAF}}(T|Y, T'|Y) + d_{\text{MAF}}(T|X - Y, T'|X - Y) - 1. \tag{2}$$

Deciding whether (1) or (2) holds, first requires the addition of a "placeholder" taxon ρ_1 into $T|Y$ and $T'|Y$ which represents the location of the $X - Y$ side of the trees. Similarly, a placeholder taxon ρ_2 is added to $T|X - Y$ and $T'|X - Y$ which represents the location of the Y side of the trees. It is then necessary to query whether there exists a maximum agreement forest for the ρ_1-augmented instance (respectively, the ρ_2-augmented instance) that isolates ρ_1 (respectively, ρ_2) in a singleton component. Depending on the outcome, a distinction can be made between (1) and (2); for full details see (Bordewich et al. 2017). Tubro answers these queries by first solving two separate ILPs for the ρ_1-augmented instance; one in which the edge entering ρ_1 is definitely cut, and one where the decision whether to cut this edge is left to the ILP. Symmetrically, it then solves two ILPs for the ρ_2-augmented instance. Hence, for a cluster Y that is common to T and T', four ILPs need to be solved in total. There are a number of additional subtle technicalities concerning the interaction of the placeholder taxon and the reduction rules from Kelk and Linz (2020) but these are out of scope of the current article.

The decomposition of one ILP into four ILPs sounds rather cumbersome, but given that the bottleneck for Tubro's performance is the number of taxa in an instance—recall that the ILP has size $O(|X|^4)$—the cluster reduction can potentially cause a significant speedup if $|Y|$ and $|X - Y|$ are roughly the same, and/or the reduction activates multiple times.

We did not switch on the cluster reduction when applying Tubro to the 89 instances that uSPR could not solve (post-kernelization) after 5 min. However, we did briefly check whether the cluster reduction might have helped with these 89 instances. We observed that 86 of the instances had no common cluster. The 3 instances that did have a common cluster had parameter combinations (t, s, k) of $(250, 50, 35)$, $(300, 50, 35)$ and $(200, 70, 35)$. For these tree pairs, we observed that the clusters were heavily imbalanced, meaning that $|Y|$ was small and $|X - Y|$ was large. This limited the practical impact of the cluster reduction since, for the larger tree pair on $X - Y$, it was still necessary to construct a prohibitively large ILP.

References

Alber, J., Betzler, N., & Niedermeier, R. (2006). Experiments on data reduction for optimal domination in networks. *Annals of Operations Research, 146*(1), 105–117.

Allen, B., & Steel, M. (2001). Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics, 5*, 1–15.

Atkins, R., & McDiarmid, C. (2019). Extremal distances for subtree transfer operations in binary trees. *Annals of Combinatorics, 23*(1), 1–26.

Bordewich, M., Scornavacca, C., Tokac, N., & Weller, M. (2017). On the fixed parameter tractability of agreement-based phylogenetic distances. *Journal of Mathematical Biology, 74*(1–2), 239–257.

Chen, J., Fan, J.-H., & Sze, S.-H. (2015). Parameterized and approximation algorithms for maximum agreement forest in multifurcating trees. *Theoretical Computer Science, 562*, 496–512.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research, 4*(3), 233–235.

Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., et al. (2015). *Parameterized algorithms* (1st ed.). Springer.

Downey, R., & Fellows, M. (2013). *Fundamentals of parameterized complexity* (Vol. 4). Springer.

Fellows, M., Jaffke, L., Király, A.I., Rosamond, F., & Weller, M. (2018). What is known about vertex cover kernelization? In H. J. Böckenhauer, D. Komm, & W. Unger (Eds.), *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated* (pp.330–356). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-98355-4_19.

Felsenstein, J. (2004). *Inferring phylogenies*. Sinauer Associates.

Ferizovic, D., Hespe, D., Lamm, S., Mnich, M., Schulz, C., & Strash, D. (2020). Engineering kernelization for maximum cut. In *2020 Proceedings of the Twenty-Second Workshop on Algorithm Engineering and Experiments (ALENEX)* (pp. 27–41). SIAM.

Fischer, M., & Kelk, S. (2016). On the maximum parsimony distance between phylogenetic trees. *Annals of Combinatorics, 20*(1), 87–113.

Fitch, W. M. (1971). Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Biology, 20*(4), 406–416.

Fomin, F., Lokshtanov, D., Saurabh, S., & Zehavi, M. (2019). *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press.

Harding, E. (1971). The probabilities of rooted tree-shapes generated by random bifurcation. *Advances in Applied Probability*, 3(1), 44–77.

Hein, J., Jiang, T., Wang, L., & Zhang, K. (1996). On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics, 71*(1–3), 153–169.

Henzinger, M., Noe, A., Schulz, C. (2020). Shared-memory branch-and-reduce for multiterminal cuts. In *2020 Proceedings of the Twenty-Second Workshop on Algorithm Engineering and Experiments (ALENEX)* (pp. 42–55). SIAM.

Hickey, G., Dehne, F., Rau-Chaplin, A., & Blouin, C. (2008). SPR distance computation for unrooted trees. *Evolutionary Bioinformatics, 4*, EBO-S419.

Huson, D., Rupp, R., & Scornavacca, C. (2011). *Phylogenetic networks: Concepts, algorithms and applications*. Cambridge University Press.

Kelk, S., & Fischer, M. (2017). On the complexity of computing mp distance between binary phylogenetic trees. *Annals of Combinatorics, 21*(4), 573–604.

Kelk, S., Fischer, M., Moulton, V., & Wu, T. (2016). Reduction rules for the maximum parsimony distance on phylogenetic trees. *Theoretical Computer Science, 646*, 1–15.

Kelk, S., & Linz, S. (2019). A tight kernel for computing the tree bisection and reconnection distance between two phylogenetic trees. *SIAM Journal on Discrete Mathematics, 33*(3), 1556–1574.

Kelk, S., & Linz, S. (2020). New reduction rules for the tree bisection and reconnection distance. *Annals of Combinatorics*, 24(3), 475–502.

Kelk, S., & Stamoulis, G. (2017). A note on convex characters, fibonacci numbers and exponential-time algorithms. *Advances in Applied Mathematics, 84*, 34–46.

Kuhner, M., & Yamato, J. (2015). Practical performance of tree comparison metrics. *Systematic Biology, 64*(2), 205–214.

Li, Z., Zeh, N. (2017). Computing maximum agreement forests without cluster partitioning is folly. In *25th Annual European Symposium on Algorithms (ESA 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

LLC Gurobi Optimization. (2020). *Gurobi optimizer reference manual*.

Mertzios, G., Nichterlein, A., & Niedermeier, R. (2020). The power of linear-time data reduction for maximum matching. *Algorithmica, 82*, 3521–3565.

Moulton, V., & Wu, T. (2015). A parsimony-based metric for phylogenetic trees. *Advances in Applied Mathematics, 66*, 22–45.
Richards, E., Brown, J., Barley, A., Chong, R., & Thomson, R. (2018). Variation across mitochondrial gene trees provides evidence for systematic error: How much gene tree variation is biological? *Systematic Biology, 67*(5), 847–860.
Semple, C., & Steel, M. (2003). *Phylogenetics*. Oxford University Press.
St John, K. (2017). The shape of phylogenetic treespace. *Systematic Biology, 66*(1), e83.
van Iersel, L., Kelk, S., & Scornavacca, C. (2016). Kernelizations for the hybridization number problem on multiple nonbinary trees. *Journal of Computer and System Sciences, 82*(6), 1075–1089.
Whidden, C., Beiko, R. G., & Zeh, N. (2013). Fixed-parameter algorithms for maximum agreement forests. *SIAM Journal on Computing, 42*(4), 1431–1466.
Whidden, C., & Matsen, F. (2018). Calculating the unrooted subtree prune-and-regraft distance. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 16*(3), 898–911.
Whidden, C., Zeh, N., & Beiko, R. G. (2014). Supertrees based on the subtree prune-and-regraft distance. *Systematic Biology, 63*(4), 566–581.
Wu, Y. (2009). A practical method for exact computation of subtree prune and regraft distance. *Bioinformatics, 25*(2), 190–196.
Yoshida, R., Fukumizu, K., & Vogiatzis, C. (2019). Multilocus phylogenetic analysis with gene tree clustering. *Annals of Operations Research, 276*(1–2), 293–313.

Authors and Affiliations

Rim van Wersch[1] · Steven Kelk[1] · Simone Linz[2] · Georgios Stamoulis[1]

✉ Georgios Stamoulis
georgios.stamoulis@maastrichtuniversity.nl

Rim van Wersch
rim@hyperboreanventures.com

Steven Kelk
steven.kelk@maastrichtuniversity.nl

Simone Linz
s.linz@auckland.ac.nz

[1] Department of Data Science and Knowledge Engineering (DKE), Maastricht University, Maastricht, The Netherlands

[2] School of Computer Science, University of Auckland, Auckland, New Zealand