The Approximability of Maximum Rooted Triplets Consistency with Fan Triplets and Forbidden Triplets

Jesper Jansson[1(✉)], Andrzej Lingas[2], and Eva-Marta Lundell[2]

[1] Laboratory of Mathematical Bioinformatics, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan
jj@kuicr.kyoto-u.ac.jp
[2] Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden
{Andrzej.Lingas,Eva-Marta.Lundell}@cs.lth.se

Abstract. The *maximum rooted resolved triplets consistency problem* takes as input a set $\mathcal{R}$ of resolved triplets and asks for a rooted phylogenetic tree that is consistent with the maximum number of elements in $\mathcal{R}$. This paper studies the polynomial-time approximability of a generalization of the problem where in addition to resolved triplets, the input may contain fan triplets and forbidden triplets. To begin with, we observe that the generalized problem admits a 1/4-approximation in polynomial time. Next, we present a polynomial-time approximation scheme (PTAS) for dense instances based on smooth polynomial integer programming. Finally, we generalize Wu's exact exponential-time algorithm in [19] for the original problem to also allow fan triplets, forbidden resolved triplets, and forbidden fan triplets. Forcing the algorithm to always output a k-ary phylogenetic tree for any specified $k \geq 2$ then leads to an exponential-time approximation scheme (ETAS) for the generalized, unrestricted problem.

Keywords: Bioinformatics · Approximation algorithms · Phylogenetic tree · Rooted triplet · Smooth integer program

1 Introduction

Phylogenetic trees are used by scientists to describe treelike evolutionary history for various kinds of objects such as biological species, natural languages, manuscripts, *etc.* [7]. Inferring an accurate phylogenetic tree from experimental data can be a difficult task; for example, computationally expensive methods like maximum likelihood that are known to yield good trees may be impractical for large data sets [6]. One potential remedy is the divide-and-conquer approach: first apply some expensive method to obtain a collection of highly reliable trees for small, overlapping subsets of the leaf labels, and then use a computationally cheaper method to merge these trees into a phylogenetic supertree [6,10,15].

JJ was funded by The Hakubi Project and KAKENHI grant number 26330014.

F. Cicalese et al. (Eds.): CPM 2015, LNCS 9133, pp. 272–283, 2015.
DOI: 10.1007/978-3-319-19929-0_23

A concept that captures the combinatorial aspects of the smallest meaningful building blocks of a phylogenetic supertree in the rooted case is *rooted triplets consistency*. Given a set $\mathcal{R}$ of possibly contradicting rooted phylogenetic trees with exactly three leaves each (so-called *rooted triplets*), the *maximum rooted triplets consistency problem* asks for a tree that contains as many of the rooted triplets in $\mathcal{R}$ as possible as embedded subtrees. Most previous work on the topic (e.g., [1,4,5,8,16,18,19]) has focused on the case where all the given rooted triplets are *resolved triplets*, meaning that they are binary. This paper considers a more general problem variant where $\mathcal{R}$ may also contain non-binary triplets (called *fan triplets*) that should preferably be included in the output tree as well as *forbidden triplets* that should be avoided.

1.1 Definitions

A (rooted) *phylogenetic tree* is a rooted, unordered tree with no internal nodes of degree 1 and whose leaves are distinctly labeled. To simplify the presentation, we identify each leaf in a phylogenetic tree with the unique element that labels it. The set of all leaf labels in a phylogenetic tree T is denoted by $\Lambda(T)$. For any $x, y \in \Lambda(T)$, $lca^T(x, y)$ is the lowest common ancestor in T of x and y.

Suppose that T is a phylogenetic tree. For any $x, y, z \in \Lambda(T)$, define the following four types of constraints on T:

1. $xy|z$, specifying that $lca^T(x, y)$ should be a proper descendant of $lca^T(x, z)$ (or equivalently, that $lca^T(x, y)$ should be a proper descendant of $lca^T(y, z)$).
2. $x|y|z$, specifying that $lca^T(x, y) = lca^T(x, z) = lca^T(y, z)$ should hold.
3. $\neg xy|z$, specifying that $lca^T(x, y)$ should not be a not proper descendant of $lca^T(x, z)$ (or equivalently, that $lca^T(x, y)$ should not be a proper descendant of $lca^T(y, z)$).
4. $\neg x|y|z$, specifying that the same node should not be the lowest common ancestor of a and b for all pairs $a, b \in \{x, y, z\}$.

The *maximum rooted triplets consistency problem* (MTC) is: given a set S of leaf labels and a set $\mathcal{R}$ of constraints as defined above, output a phylogenetic tree T with $\Lambda(T) = S$ that satisfies as many constraints from $\mathcal{R}$ as possible. In this paper, the special case of MTC where all constraints in $\mathcal{R}$ are of type 1 is called the *maximum rooted resolved triplets consistency problem* (MRTC), and the special case where all constraints are of type 1 or type 3 is called the *maximum mixed rooted resolved triplets consistency problem* (MMRTC).[1]

To express the size of an instance of MTC, we write $n = |S|$ and $m = |\mathcal{R}|$. An instance $(S, \mathcal{R})$ of MTC is *complete* if, for every $S' \subseteq S$ with $|S'| = 3$, $\mathcal{R}$ contains at least one constraint involving the three elements in S' only. It is called *dense* if it contains $\Omega(n^3)$ constraints. Note that any complete instance is dense.

[1] MRTC is called MAX-LEVEL-0 in [4], MaxRTC in [5], MILCT in [8,12], MaxCL-0-Dense in [11], MTC in [16], and MCTT in [18,19]. MMRTC is called MMTT in [9].

Remark 1. Phylogenetic trees with exactly three leaves are commonly referred to as *rooted triplets* in the literature. A rooted triplet t is either a binary or a non-binary tree. In the former case, t is a *resolved triplet* and always satisfies a constraint of type 1, and if this constraint is also satisfied in a phylogenetic tree T then t and T are said to be *consistent*. Similarly, if t is non-binary then t is called a *fan triplet* and always satisfies a constraint of type 2; if it is also satisfied in a phylogenetic tree T then t and T are *consistent*. Thus, an equivalent formulation of MTC is: given two sets $\mathcal{C}$ and $\mathcal{F}$ of rooted triplets, output a phylogenetic tree T with $\Lambda(T) = \bigcup_{t \in \mathcal{C} \cup \mathcal{F}} \Lambda(t)$ maximizing $|T(\mathcal{C})| - |T(\mathcal{F})|$, where $T(\mathcal{X})$ for any set $\mathcal{X}$ of rooted triplets is the subset of $\mathcal{X}$ consistent with T. In analogy with this terminology, constraints of type 1, 2, 3, and 4 are called *resolved triplets*, *fan triplets*, *forbidden resolved triplets*, and *forbidden fan triplets* from here on.

1.2 Previous Results

Aho *et al.* [1] presented a polynomial-time algorithm that determines if there exists a phylogenetic tree consistent with *all* of the resolved triplets in a given set, and if so, outputs such a tree. Its time complexity was improved from $O(mn)$ to $\min\{O(n + mn^{1/2}), O(m + n^2 \log n)\}$ by Henzinger *et al.* [10]. He *et al.* [9] extended Aho *et al.*'s algorithm to the case where the input also contains forbidden resolved triplets, and the resulting running time to determine if there exists a phylogenetic tree that satisfies all the input constraints is $O((m + n)n \log n)$.

In comparison, the optimization versions of rooted triplets consistency turn out to be harder. MRTC is NP-hard [3,12,19], even if restricted to dense problem instances [11]. Furthermore, MRTC in the non-dense case is APX-complete [4]. The supplementary version of MRTC in which the objective is to remove as few elements as possible from the input $\mathcal{R}$ so that there exists a phylogenetic tree consistent with the resulting $\mathcal{R}$ is $W[2]$-hard and cannot be approximated within $c \ln n$ for some constant $c > 0$ in polynomial time, unless P = NP [5]. As for positive results for MRTC, Gąsieniec *et al.* [8] presented a top-down, polynomial-time 1/3-approximation algorithm, and Wu [19] gave a bottom-up, polynomial-time heuristic that was shown experimentally to perform well in practice. Byrka *et al.* [5] later modified Wu's heuristic to guarantee that it too achieves an approximation ratio of 1/3. Other heuristics for MRTC (with unknown approximation ratios) have been published in [16,18]. An exact algorithm for MRTC running in $O(3^n(m + n^2))$ time and $O(2^n)$ space was given by Wu in [19]. Finally, we remark that the 1/3-approximation algorithm for MRTC in [8] was generalized to a polynomial-time 1/3-approximation algorithm for MMRTC in [9].

The unrooted analogue of a resolved triplet, called a *quartet* [17], is an unrooted tree with two internal nodes and four distinctly labeled leaves. The corresponding maximum quartets consistency problem is MAX SNP-hard [14,17], but the complete version of the problem admits a PTAS [14]. In an unpublished manuscript [13], we have outlined how to obtain a similar PTAS for dense MRTC.

See the survey in Sect. 2 in [5] for references to other rooted triplets consistency-related problems in the literature involving enumeration, ordered trees, phylogenetic networks, multi-labeled phylogenetic trees (MUL-trees), etc.

1.3 Our Contributions

We first show how any known polynomial-time 1/3-approximation algorithm for MRTC (e.g., [5,8]) can be applied to obtain a polynomial-time 1/4-approximation algorithm for MTC (Sect. 2).

The APX-completeness of MRTC [4] (and hence, MTC) rules out the possibility of finding a PTAS for MTC in the general case. Nevertheless, we make further progress on the approximation status of MTC by presenting a PTAS for MTC restricted to *dense* instances based on smooth polynomial integer programming, using some ideas from [14] and generalizing our unpublished work in [13] (Sect. 3).

Next, we extend Wu's exact exponential-time algorithm for MRTC [19] to MTC (Sect. 4). We let the algorithm take an additional parameter $k \geq 2$ as input and force the output to be a phylogenetic tree in which every internal node has at most k children. The resulting algorithm runs in $O(2^{(n+1)\log_2(k+1)}(m+n))$ time. This may be $\Omega(n^n)$ if k is unrestricted, but the running time is single-exponential in n when $k = O(1)$, and we use this fact to design an exponential-time approximation scheme (ETAS) for MTC with no restrictions on k.

Finally, we describe how to adapt our algorithms to the weighted case, where nonnegative weights are assigned to the triplet constraints and the objective is to construct a phylogenetic tree that maximizes the sum of the weights of the satisfied constraints (Sect. 5). In case of our PTAS and our ETAS, we have to additionally assume that the ratio between the largest and the smallest constraint weights is bounded by a constant.

2 A 1/4-Approximation Algorithm for MTC

The maximum rooted resolved triplets consistency problem (MRTC) admits a 1/3-approximation algorithm running in polynomial time [5,8]. The algorithms in [5,8] always output a binary tree, so they also yield (at least) a 1/3-approximation when in addition to resolved triplets, forbidden fan triplets are included in the input. We use this fact to design a 1/4-approximation algorithm for the maximum rooted triplets consistency problem (MTC) as follows.

Algorithm 1
Input: A set $\mathcal{R}$ of m triplet constraints over an n-element set S.
Output: A phylogenetic tree with n leaves distinctly leaf-labeled by S.

1. If $\mathcal{R}$ contains at least $m/4$ fan triplets and forbidden resolved triplets then output a tree whose root has n children, each of them a leaf with a distinct label in S, and stop.
2. Extract the set $\mathcal{R}'$ of all resolved triplets and forbidden fan triplets from $\mathcal{R}$ and apply any known polynomial-time 1/3-approximation algorithm for MRTC (e.g., [5] or [8]) to $\mathcal{R}'$. Output the tree produced by the latter.

Theorem 1. *Algorithm 1 is a polynomial-time 1/4-approximation algorithm for MTC.*

Proof. We need to show that the algorithm outputs a phylogenetic tree satisfying at least 1/4 of the input triplet constraints. There are two cases:

If $\mathcal{R}$ contains at least $m/4$ fan triplets and forbidden resolved triplets then the star phylogenetic tree output in the first step satisfying all the fan triplets and all the forbidden resolved triplets satisfies at least $m/4$ input triplet constraints.

Otherwise, $\mathcal{R}$ contains at least $3m/4$ resolved triplets and forbidden fan triplets. The 1/3-approximation algorithm run on them in the second step yields a phylogenetic tree satisfying at least $\frac{1}{3} \cdot \frac{3m}{4} = m/4$ input triplet constraints. □

3 A PTAS for Dense MTC

Analogously to [14] for the unrooted case, we first show that any rooted phylogenetic tree T with a leaf label set $S = \Lambda(T)$ can be represented approximately by a *decomposition tree* consisting of:

1. a bounded-size subtree (termed *kernel*) K of T on non-leaf nodes, and
2. subsets of S (forming a partition of S) in one-to-one correspondence with the leaves of K, where the elements of each subset are children of the corresponding leaf of K.

In particular, an optimal tree T_{opt} for a given instance of MTC can be approximately represented by such a decomposition tree which preserves enough of the original triplet constraints to serve as a good approximation. More precisely, the number of input triplet constraints satisfied by the approximate tree differs from that of T_{opt} by an arbitrarily small fraction, depending on the number of subsets in the partition. We find an approximate solution by enumerating all possible kernels, and for each one, finding the approximately best partition of S.

Recall that an instance of MTC is *dense* if the input set of triplet constraints has $\Omega(n^3)$ elements. The analysis of the accuracy of our approximate solution relies on the fact that for a dense instance, the number of input triplet constraints satisfied by T_{opt} is $\Omega(n^3)$, since it is at least 1/4 of the number of the constraints by Theorem 1.

Let k be a fixed integer, and let $S_1, S_2, ..., S_k$ be a partition of the set S. A subset S_i is termed a *bin*. For each bin S_i, there is a non-leaf node of degree $|S_i| + 1$ in the decomposition tree, termed a *bin root*, connected by an edge to each element in the bin. Algorithm 2, given below, transforms an input tree into its decomposition tree by joining adjacent subtrees of T until the bin is large enough, for some given maximum bin size b. If a bin is smaller than $b/2$, and there is another bin also smaller than $b/2$ in an adjoining subtree, the two small bins may be joined into one single bin. The resulting kernel K is the subtree of the output decomposition tree induced by remaining non-leaf nodes, with the subtrees defining the bins removed. The output decomposition tree T_k consists of the kernel K, with the bin roots as leaves of K, and the elements in each bin being children of its respective bin root.

Algorithm 2. k-bin decomposition(T)
Input: A phylogenetic tree T with n leaves.
Output: A decomposition tree T_k of T.

- Traverse T, and for every node v visited, check if the size of the subtree $T(v)$ of T rooted at v is less or equal to $6n/k$. If so, v is denoted a bin root (unless v is a leaf), and all internal edges of $T(v)$ except for edges incident to a leaf are contracted, so that $T(v)$ becomes a tree of height 1. If the size of $T(v)$ is larger than $6n/k$, continue traversing T at a child of v.
- For a single leaf l that is not in a bin, the edge between l and its parent is subdivided to create a new bin root associated with l.
- A bin of size $\leq 3n/k$ is *small*. Let b be a small bin, and let v be the parent of b. If another small bin b' exists as a child of a sibling of v, b and b' are combined to a single bin.

Lemma 1. *Algorithm 2 for k-bin decomposition produces a decomposition tree T_k having at most k bins, where each bin is of size less or equal to $6n/k$.*

Proof. In Lemma 1 in [14], a proof of an analogous lemma for quartets is given. The reader is referred to this proof for more details.

As a consequence of the decomposition procedure, the number of bins will be bounded by k since the merging of small bins in the third step guarantees that there are not too many small bins. Lemma 1 in [14] shows that the number of small bins is strictly smaller than twice the number of large bins. Since a large bin has a size of at least $3n/k$, the number of large bins is at most $k/3$. Let the number of large bins be l, and the number of small bins be s. Then the total number of bins is $s + l < l + 2l = 3l < 3 \cdot k/3 = k$. So, T_k has less than k bins, each of size at most $6n/k$. □

Let R be the input set of triplet constraints. For any phylogenetic tree T, let R_T denote the subset of triplet constraints in R that are satisfied by T.

Since the decomposition algorithm works by contracting some edges of T_{opt} and transferring leaves to neighboring bins, it follows that for any triplet $\{a, b, c\}$ where a, b and c are in different bins, $ab|c \in R_{T_k}$ if and only if $ab|c \in R_{T_{opt}}$, $\neg ab|c \in R_{T_k}$ if and only if $\neg ab|c \in R_{T_{opt}}$, and similarly, $a|b|c \in R_{T_k}$ if and only if $a|b|c \in R_{T_{opt}}$, and $\neg a|b|c \in R_{T_k}$ if and only if $\neg a|b|c \in R_{T_{opt}}$.

Lemma 2. *The tree T_k that is a k-bin decomposition of T_{opt} satisfies $|R_{T_k} \cap R| \geq |R_{T_{opt}} \cap R| - \frac{c}{k} \cdot n^3$ input triplet constraints, for some constant c.*

Proof. Any triplet topology in $R_{T_{opt}} \setminus R_{T_k}$ must have two or more leaves in the same bin. The number of such triplet topologies with three or two leaves in the same bin is at most $1/6\,(6n/k)^3\,k \;+\; 1/2\,(6n/k)^2\,nk \leq 24n^3/k$ for $k \geq 6$. Each of the above triplet topologies may contribute to at most four triplet constraints in R (one fan triplet and three forbidden resolved triplets in the worst case). Hence, assuming that $k \geq 6$, we have $|R_{T_k} \cap R| \geq |R_{T_{opt}} \cap R| - 96n^3/k$. □

Label-to-bin Assignment: Suppose that we are given a kernel K with at most k leaves of a hypothetical phylogenetic tree distinctly leaf-labeled by S. The *Label-to-Bin Assignment problem* (LBA) for a set R of triplet constraints asks for an assignment of labels in S to at most k bins of size $\leq 6n/k$ that

completes K to T_k and maximizes $|R \cap R_{T_k}|$. The supertree of K induced by such an assignment is called a *completion of* K.

Jiang *et al.* [14] showed that although the corresponding LBA problem for unrooted quartets is NP-hard, it admits a PTAS relying on a modified PTAS for smooth polynomial integer programs by Arora *et al.* [2]. We adapt this technique to our problem. First, for every resolved triplet $ab|c$ in R, define the polynomial:

$$p_{ab|c}(x) = \sum_{ij|k \in R_{T_k}} x_{ai}x_{bj}x_{ck} + x_{bi}x_{aj}x_{ck}$$

Here, the term $x_{sb} = 1$ if label s is assigned to bin b, and 0 otherwise. Next, for every fan triplet $a|b|c$ in R, define the following polynomial, where $Per(a, b, c)$ stands for the set of all one-to-one mappings from $\{a, b, c\}$ to $\{a, b, c\}$:

$$p_{a|b|c}(x) = \sum_{i|j|k \in R_{T_k}} \sum_{\delta \in Per(a,b,c)} x_{\delta(a)i}x_{\delta(b)j}x_{\delta(c)k}$$

For every forbidden resolved triplet $\neg ab|c$ in R, define the polynomial:

$$p_{\neg ab|c}(x) = p_{ac|b}(x) + p_{bc|a}(x) + p_{a|b|c}(x)$$

Similarly, for every forbidden fan triplet $\neg a|b|c$ in R, define the polynomial:

$$p_{\neg a|b|c}(x) = p_{ab|c}(x) + p_{ac|b}(x) + p_{bc|a}(x)$$

Finally, define:

$$p(x) = \sum_{ab|c \in R} p_{ab|c}(x) + \sum_{a|b|c \in R} p_{a|b|c}(x) + \sum_{\neg ab|c \in R} p_{\neg ab|c}(x) + \sum_{\neg a|b|c \in R} p_{\neg a|b|c}(x)$$

The optimization problem becomes: Maximize $p(x)$ subject to $\sum_{i=1}^{k} x_{si} = 1$ for each leaf s, and $\sum_{s=1}^{n} x_{si} \leq 6n/k$ for each bin i. (The first condition ensures that each label is assigned to exactly one bin and the second condition maintains the k-bin property.) Our polynomial integer program is an *$O(1)$-smooth degree-3 polynomial integer program* according to the following definition from [2]: *An $O(1)$-smooth degree-d polynomial integer program is to maximize $p(x_1, ..., x_n)$ subject to $x_i \in \{0, 1\}$, $\forall i \leq n$, where $p(x_1, ..., x_n)$ is a degree- d polynomial in which the coefficient of each degree-i monomial (term) is $O(n^{d-i})$.*

Lemma 3. *(Arora* et al.*[2]) Let m be the maximum value of an $O(1)$-smooth degree-d polynomial integer program $p(x_1, ..., x_n)$. For each $\epsilon > 0$, there is a polynomial-time algorithm that finds a 0/1 assignment α for the x_i satisfying $p(\alpha(x_1), ..., \alpha(x_n)) \geq m - \epsilon n^d$.*

The PTAS of Arora *et al.* first solves the fractional version of the problem. It then rounds the obtained fractional value for each variable individually in order to obtain an integer solution. However, this is not possible in our case because of the condition $\sum_{i=1}^{k} x_{si} = 1$ for each leaf s. Instead, following [14], we set $x_{si} = 1$ and $x_{sj} = 0$ for $j \neq i$ with probability equal the fractional value for x_{si}. In effect, exactly one of the variables $x_{s1}, ..., x_{sk}$ is set to 1 and the rest to 0. In analogy to Theorem 2.6 in [14], we obtain the next lemma.

Lemma 4. *For each $\epsilon > 0$, there is a polynomial-time algorithm which, for each instance of the LBA specified by a set R of triplet constraints for dense MTC and a kernel K, produces a completion T' of K such that $|R_{T'} \cap R| \geq |R_{\hat{T}} \cap R| - \epsilon n^3$, where $\hat{T}$ is an optimal completion of K.*

T_{opt} can be decomposed into a kernel with at most k leaves and k bins of size $\leq 6n/k$ (i.e., the tree T_k) as shown in Lemmas 1 and 2. Given any input set of triplet constraints, for each kernel with k leaves, an approximate optimal assignment of leaves to bins of such size can be found in polynomial time by Lemma 4. Hence, dense MTC can be approximated in the following way:

Theorem 2. *For each $\epsilon > 0$, there is a polynomial-time algorithm which, for each instance R of dense MTC, produces a tree T_k that approximates T_{opt} in such a way that $|R_{T_k} \cap R| \geq (1 - \epsilon)|R_{opt} \cap R|$.*

Proof. Let c be the constant specified in Lemma 2. By Lemmas 2 and 4, $|R_{T_k} \cap R| \geq |R_{T_{opt}} \cap R| - (c/k + \epsilon') \cdot n^3 \geq (1 - c/(c'k) - \epsilon'/c')|R_{T_{opt}} \cap R|$, where c' is a constant satisfying $|R_{T_{opt}} \cap R| \geq c'n^3$. We estimate this constant by the density of R and Theorem 1. By picking $k \geq \frac{2c}{c'\epsilon}$ and $\epsilon' \leq \frac{c'\epsilon}{2}$, we obtain the theorem. □

4 An ETAS for MTC

The following additional notation will be used. For any node u in a phylogenetic tree T with a leaf label set $S = \Lambda(T)$, let S_u be the subset of S labeling the leaves of the subtree rooted at u. For any node v of T, let P_v be the partition of S_v into $S_{v_1}, ..., S_{v_l}$, where $v_1, ..., v_l$ are the children of v.

For a partition P of $U \subseteq S$ into l subsets, let $w_2(P)$ be the number of resolved triplets $ab|c$ such that a and b belong to two distinct subsets in P and $c \notin U$. Similarly, let $w_3(P)$ be the number of fan triplets $a|b|a$ such that a, b, c belong to three different subsets in P. Next, let $w_{f2}(P)$ be the number of forbidden resolved triplets $\neg ab|c$ such that a and c belong to two distinct subsets in P and $b \notin U$, or b and c belong to two distinct subsets in P and $a \notin U$, or a, b, c belong to three different subsets in P. Finally, let $w_{f3}(P)$ be the number of forbidden fan triplets $\neg a|b|c$ such that two elements in $\{a, b, c\}$ belong to two distinct subsets in P and the remaining one does not belong to any of the subsets. We have:

Lemma 5. *Given a partition P of $U \subseteq S$ into l subsets, $w_2(P)$, $w_3(P)$, $w_{f2}(P)$ and $w_{f3}(P)$ can be computed in $O(m+n)$ time, where m is the number of input triplet constraints and n is the size of S.*

Proof. We "color" the elements in U with l colors according to P and the elements in $S \backslash U$ with another color, and then examine each input triplet constraint to check if it increases $w_2(P)$, $w_3(P)$, $w_{f2}(P)$, or $w_{f3}(P)$ by one. □

Remark 2. When $l = 2$ in Lemma 5, $w_2(P)$ is the same as $w(V_1, V_2)$ in Wu's exact algorithm for MRTC [19]. Theorem 2 in [19] computes $w(V_1, V_2)$ in $O(m + n^2)$ time, so using our Lemma 5 instead slightly improves the running time of Wu's algorithm from $O(3^n(m + n^2))$ to $O(3^n(m + n))$.

Lemma 6. *For a phylogenetic tree T with leaves labeled with elements in S, the number of input triplet constraints consistent with T is equal to* $\sum_{v \in T}(w_2(P_v) + w_3(P_v) + w_{f2}(P_v) + w_{f3}(P_v))$.

We now analyze how much is lost by forcing the solution to an instance of MTC to be a *k-ary phylogenetic tree*, defined as a phylogenetic tree in which every internal node has degree at most k, where k is any integer such that $k \geq 2$:

Theorem 3. *For any phylogenetic tree T, there exists a k-ary phylogenetic tree T' with $\Lambda(T') = \Lambda(T)$ that satisfies at least a fraction of $(1 - 12/k)$ of the input triplet constraints satisfied by T.*

Proof. We shall replace each node v of T having more than k children by a subtree in which all nodes have at most k children. Let $v_1,...,v_l$ be the children of v. Note that $l > k$. To start with, assign to each forbidden resolved triplet $\neg ab|c$ contributing to $w_{f2}(P_v)$, either the resolved triplet $ac|b$, where a and c belong to distinct S_{v_i}, S_{v_j} and $b \notin S_v$, or the resolved triplet $bc|a$, where a and c belong to distinct S_{v_i}, S_{v_j} and $a \notin S_v$, or the fan triplet $a|b|c$, where all a, b, c belong to three distinct S_{v_i}, S_{v_j}, S_{v_q}. Similarly, assign to each forbidden fan triplet $\neg a|b|c$ contributing to $w_{f3}(P_v)$, either the resolved triplet $ab|c$, where a and b belong to distinct S_{v_i}, S_{v_j} and $c \notin S_v$, or the resolved triplet $ac|b$, where a and c belong to distinct S_{v_i}, S_{v_j} and $b \notin S_v$, or the resolved triplet $bc|a$, where a and c belong to distinct S_{v_i}, S_{v_j} and $a \notin S_v$. Let $f_2(P_v)$ be the cardinality of the multiset of assigned resolved triplets and let $f_3(P_v)$ be the cardinality of the multiset of assigned fan triplets. Then $w_{f2}(P_v) + w_{f3}(P_v) = f_2(P_v) + f_3(P_v)$.

For the sake of the proof, partition the family of subsets $S_{v_1}, ..., S_{v_l}$ into k groups uniformly at random. Consider any fan triplet $a|b|c$ contributing to $w_3(P_v)$ (i.e., having each of its elements in a distinct S_{v_i}) or to $f_3(P_v)$ (i.e., being assigned to a forbidden resolved triplet). The probability that any two elements in $\{a, b, c\}$ fall into the same group is bounded from above by $1/k + 2/k \leq 3/k$. Hence, there exists a partition of the family of $S_{v_1}, ..., S_{v_l}$ into k groups such that at least a $(1 - 3/k)$ fraction of triples $a|b|c$ contributing to $w_3(P_v) + f_3(P_v)$ will have all its elements in three different groups. For each group g in the latter partition, first construct an arbitrary rooted resolved tree F_g whose leaves are labeled by the children v_i of v for which $S_{v_i} \in g$ and then replace each leaf labeled by v_i in F_g by the subtree of T rooted at v_i. Next, delete the edges in T connecting v with its children and instead connect v to the roots of the trees F_g by edges. Observe that the same fan triplet may contribute to $w_3(P_v)$ and it may also contribute up to three times to $f_3(P_v)$, (i.e., it may be assigned to up to three forbidden triplets contributing to $w_{f2}(P_v)$). It follows that the sum of the new value of $w_3(P_v) + f_3(P_v)$ (provided that we keep the same assignments if possible) is at least $(1-4 \cdot 3/k)$ of the sum of the previous value of $w_3(P_v)+f_3(P_v)$.

In turn, consider any resolved triplet $ab|c$ that contributes to $w_2(P_v)$ or to $f_2(P_v)$ (i.e., is assigned to a forbidden resolved triplet contributing to $w_{f2}(P_v)$ or a forbidden fan triplet contributing to $w_{f3}(P_v)$). After the transformation of T, the following holds: If the labels a and b belong to subsets in the same group g then $ab|c$ can neither contribute to $w_2(P_v)$ nor to $f_2(P_v)$ (i.e., to be assigned to

a forbidden resolved triplet contributing to $w_{f2}(P_v)$ or to a forbidden fan triplet contributing to $w_{f3}(P_v)$). On the other hand, there must exist a non-leaf node u of the binary tree F_g for which $ab|c$ correspondingly contributes to $w_2(P_u)$, or it can be assigned to the same forbidden resolved triplets now contributing to $w_{f2}(P_u)$, or it can be assigned to the same forbidden fan triplets now contributing to $w_{f3}(P_u)$. Thus, by extending the notation $f_2(\ )$ to include $f_2(P_t)$, the sum of $w_2(P_t) + f_2(P_t)$ over the tree nodes t does not change. The theorem follows from Lemma 6. □

Motivated by Theorem 3, our new approximation algorithm in this section constructs a k-ary phylogenetic tree consistent with the maximum possible number of input triplet constraints for some suitable value of k. For this purpose, we generalize Wu's algorithm [19] for MRTC which always outputs a *binary* phylogenetic tree, i.e., corresponding to the special case $k = 2$. We also need to extend Wu's algorithm to allow not only resolved triplets in the input.

Our new algorithm works as follows. For each non-singleton subset U of S, define $score(U)$ recursively by $score(U) = \max_{l=2}^{k} score_l(U)$, where $score_l(U) =$

$$\max_{l-\text{partition } U_1,..,U_l \text{ of } U} \sum_{i=1}^{l} score(U_i) + \sum_{j=2}^{3} w_j(U_1,..,U_l) + w_{fj}(U_1,...,U_l)$$

For a singleton U, $score(U)$ is set to 0. As in Wu's algorithm [19], $score(U)$ is evaluated in non-decreasing order of the sizes of subsets U of S. Then, the output phylogenetic tree is constructed by a traceback, starting from $score(S)$, and picking an l-partition of the current subset U that yields the maximum value of $score(U)$. The corresponding node of the constructed tree gets l children in one-to-one correspondence with the subsets of U forming the selected partition.

It follows by induction on $|U|$ and Lemma 6 that $score(U)$ equals the maximum number of input triplets that can be satisfied by a k-ary subtree leaf-labeled by U. This yields the optimality of the tree constructed during the traceback.

There are $\binom{n}{q}$ subsets U of S with q elements. The number of l-partitions of a subset U with q elements is l^q. Therefore, the total number of subsets partitions processed by our algorithm is $\sum_{q=1}^{n} \binom{n}{q} \sum_{\ell=2}^{k} \ell^q \leq \sum_{\ell=2}^{k} (\ell+1)^n \leq (k+1)^{n+1}$ by binomial expansion. Finally, by Lemma 5, for a given partition P of $U \subseteq S$ into l subsets, the weights $w_2(P)$, $w_3(P)$, $w_{f2}(P)$ and $w_{f3}(P)$ can be computed in $O(m+n)$ time, where m is the number of input triplet constraints and n is the size of S. We conclude that our algorithm runs in $O((k+1)^{n+1}(m+n))$ time, i.e., in $O(2^{(n+1)\log_2(k+1)}(m+n))$ time.

Theorem 4. *Let S be a set of n distinct labels and let k be an integer greater than 1. For any set R of m (resolved or forbidden resolved or fan or forbidden fan) triplet constraints on S, one can find a k-ary phylogenetic tree T with $\Lambda(T) = S$ that maximizes the number of satisfied triplet constraints in R among all k-ary phylogenetic trees in $O(2^{(n+1)\log_2(k+1)}(m+n))$ time.*

By combining Theorems 3 and 4, we obtain an exponential-time approximation scheme (ETAS) for the maximum rooted triplets consistency problem:

Theorem 5. *Let S be a set of n distinct labels and let $\epsilon > 0$ be a constant. For any set R of m (resolved or forbidden resolved or fan or forbidden fan) triplet constraints on S, one can find a phylogenetic tree T with $\Lambda(T) = S$ in $O(2^{(n+1)\log_2(\lceil 12/\epsilon \rceil + 1)}(m+n))$ time satisfying at least $(1-\epsilon)$ of the maximum number of triplet constraints in R that can be satisfied in any phylogenetic tree.*

5 Extensions to the Weighted Case

Having input triplet constraints in the form of rooted triplets and forbidden rooted triplets, it is natural to assign nonnegative real weights to them. Note that $\neg a|b|c$ is equivalent to the conjunction of $\neg ab|c$, $\neg ac|b$ and $\neg bc|a$. Consequently, MTC generalizes to the *maximum weighted rooted triplet consistency problem* (MWTC), where the objective is to construct a phylogenetic tree that maximizes the total weight of the satisfied input triplet constraints.

By Theorem 4 in [8], the 1/3-approximation algorithm for MRTC in [8] works for the weighted version of MRTC as well. Hence, the 1/4-approximation algorithm for MTC in Sect. 2 immediately generalizes to MWTC by considering sums of weights of input triplet constraints belonging to the appropriate subsets instead of just the cardinalities of the subsets. Our exact algorithm for MTC in Sect. 4 similarly generalizes to MWTC by considering sums of the weights of the respective triplet constraints instead of their numbers. However, the situation is a bit more subtle for our PTAS for dense MTC in Sect. 3 and our ETAS for MTC in Sect. 4. Because of Lemma 2 and Theorem 3, respectively, where in both cases some fraction of the triplet constraints may be lost, we need to assume that the maximum triplet constraint weight is at most $O(1)$ times larger than the minimum one in order to generalize both approximation schemes to the weighted case. Furthermore, in our PTAS, the polynomials in one-to-one correspondence with the input triplet constraints in the definition of the integer program have to be multiplied by the weight of the corresponding constraint.

6 Final Remarks

MTC is APX-complete by the APX-completeness of MRTC [4] and Theorem 1. An open problem is to improve the polynomial-time approximation ratios 1/3 and 1/4 for MRTC and MTC; by applying the technique in Sect. 2, an f-approximation for the former would give an $\frac{f}{1+f}$-approximation for the latter.

References

1. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. **10**(3), 405–421 (1981)
2. Arora, S., Karger, D., Karpinski, M.: Polynomial time approximation schemes for dense instances of NP-hard problems. J. Comput. Syst. Sci. **58**(1), 193–210 (1999)

3. Bryant, D.: Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis. Ph.D. thesis. University of Canterbury, Christchurch, New Zealand (1997)
4. Byrka, J., Gawrychowski, P., Huber, K.T., Kelk, S.: Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks. J. Discrete Algorithms **8**(1), 65–75 (2010)
5. Byrka, J., Guillemot, S., Jansson, J.: New results on optimizing rooted triplets consistency. Discrete Appl. Math. **158**(11), 1136–1147 (2010)
6. Chor, B., Hendy, M., Penny, D.: Analytic solutions for three taxon ML trees with variable rates across sites. Discrete Appl. Math. **155**(6–7), 750–758 (2007)
7. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates Inc., Sunderland (2004)
8. Gąsieniec, L., Jansson, J., Lingas, A., Östlin, A.: On the complexity of constructing evolutionary trees. J. Comb. Optim. **3**(2–3), 183–197 (1999)
9. He, Y.J., Huynh, T.N.D., Jansson, J., Sung, W.-K.: Inferring phylogenetic relationships avoiding forbidden rooted triplets. J. Bioinform. Comput. Biol. **4**(1), 59–74 (2006)
10. Henzinger, M.R., King, V., Warnow, T.: Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. Algorithmica **24**(1), 1–13 (1999)
11. van Iersel, L., Kelk, S., Mnich, M.: Uniqueness, intractability and exact algorithms: reflections on level-k phylogenetic networks. J. Bioinform. Comput. Biol. **7**(4), 597–623 (2009)
12. Jansson, J.: On the complexity of inferring rooted evolutionary trees. In: Proceedings of the Brazilian Symposium on Graphs, Algorithms, and Combinatorics (GRACO 2001). Electronic Notes in Discrete Mathematics, vol. 7, pp. 50–53. Elsevier (2001)
13. Jansson, J., Lingas, A., Lundell, E.-M.: A triplet approach to approximations of evolutionary trees. Poster H15 presented at RECOMB 2004 (2004)
14. Jiang, T., Kearney, P., Li, M.: A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. SIAM J. Comput. **30**(6), 1942–1961 (2001)
15. Kearney, P.: Phylogenetics and the quartet method. In: Jiang, T., Xu, Y., Zhang, M.Q. (eds.) Current Topics in Computational Molecular Biology, pp. 111–133. The MIT Press, Massachusetts (2002)
16. Snir, S., Rao, S.: Using max cut to enhance rooted trees consistency. IEEE/ACM Trans. Comput. Biol. Bioinf. **3**(4), 323–333 (2006)
17. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. J. Classif. **9**(1), 91–116 (1992)
18. Wu, B.Y.: Constructing evolutionary trees from rooted triplets. J. Inf. Sci. Eng. **20**, 181–190 (2004)
19. Wu, B.Y.: Constructing the maximum consensus tree from rooted triples. J. Comb. Optim. **8**(1), 29–39 (2004)