Quartets MaxCut: A Divide and Conquer Quartets Algorithm

Sagi Snir and Satish Rao

Abstract—Accurate phylogenetic reconstruction methods are currently limited to a maximum of few dozens of taxa. Supertree methods construct a large tree over a large set of taxa, from a set of small trees over overlapping subsets of the complete taxa set. Hence, in order to construct the tree of life over a million and a half different species, the use of a supertree method over the product of accurate methods, is inevitable. Perhaps the simplest version of this task that is still widely applicable, yet quite challenging, is quartet-based reconstruction. This problem lies at the root of many tree reconstruction methods and theoretical as well as experimental results have been reported. Nevertheless, dealing with false, conflicting quartet trees remains problematic. In this paper, we describe an algorithm for constructing a tree from a set of input quartet trees even with a significant fraction of errors. We show empirically that conflicts in the inputs are handled satisfactorily and that it significantly outperforms and outraces the Matrix Representation with Parsimony (MRP) methods that have previously been most successful in dealing with supertrees. Our algorithm is based on a divide and conquer algorithm where our divide step uses a semidefinite programming (SDP) formulation of MaxCut. We remark that this builds on previous work of ours [29] for piecing together trees from rooted triplet trees. The recursion for *unrooted* quartets, however, is more complicated in that even with completely consistent set of quartet trees the problem is NP-hard, as opposed to the problem for triples where there is a linear time algorithm. This complexity leads to several issues and some solutions of possible independent interest.

Index Terms—Phylogenetic reconstruction, quartets, MaxCut, supertree.

1 Introduction

The study of evolution and the construction of phylogenetic (evolutionary) trees (or phylogenies) are classical subjects in biology. DNA sequences from a variety of organisms are rapidly accumulating, providing large amounts of data to a number of sequence-based approaches for phylogenetic trees reconstruction. The goal behind the "tree of life" project is to accurately construct the tree representing the evolutionary history of over a million and a half different species. This task cannot be achieved by today's substitution-based phylogenetic reconstruction methods. Therefore, the need to design methods capable to amalgamate data taken from different sources is emerging.

Phylogenetic reconstruction methods are broadly divided into two high-level types of reconstruction methods: Sequence based and topology based. In sequence based, the input is a set of homologous sequences from different species. The method builds a phylogeny that tries to represent the evolutionary history of these sequences. However, in many cases such a set of homologues for the set of species under study does not exist. Moreover, as the accuracy of the obtained phylogeny decreases with increasing size (number of species and distance between species) [14], [21], [13], an alternative is to build small trees with very accurate sequence-based methods (e.g., Maximum Likelihood) and later amalgamate these into a big complete tree that represents each of the input small trees. This is the task of a topology-based reconstruction algorithm.

We distinguish between *rooted* and *unrooted* phylogenetic trees. In unrooted trees, the decision problem of whether there exists a tree that satisfies (or consistent with) all the input subtrees was shown to be NP-hard by Steel [31] (see Section 2.1 for an exact definition). However, in the same paper, it is shown that if the trees are rooted then this problem is solvable in polynomial time by a simple divide and conquer algorithm of Aho et al. [1]. This makes the rooted settings computationally attractive. However, a rooted setting is problematic due to the fact that the knowledge of the root is sometimes absent and an incorrect root estimation can lead to an erroneous tree. For that reason, despite its computational drawback, people resort to an unrooted setting. A quartet tree is the smallest informative unrooted tree (see Fig. 1). This, along with the fact that small trees are easier to infer than larger ones, make quartets-based methods attractive for large-scale phylogenetic reconstruction [5], [33], [32], [26], [34], [4], [14], [15]. In [14], it is shown that using only number of sites logarithmic in the number of taxa, all quartet trees in a set called *the short quartets* are inferred correctly with high probability. They subsequently combine their set of quartet trees to a complete tree using some inference rules. The work in [13] enhances this algorithm by introducing a confidence check to each quartet examined. A quartet is considered as correct only if it has sufficient confidence. The algorithm grows subtrees of increasing radii around every taxon and joins two subtrees when they share internal edges. The algorithm returns a forest where each edge in the forest is of high confidence. This algorithm alleviates the necessity in [14] of bounding the edge length as it defines a local check for each quartet separately.

- *S. Snir is with the Institute of Evolution, University of Haifa, Mount Carmel, Haifa 31905, Israel. E-mail: ssagi@research.haifa.ac.il.*
- *S. Rao is with the Computer Science Division, 681 Soda Hall, University of California, Berkeley, CA 94720-1776. E-mail: satishr@cs.berkeley.edu.*

Manuscript received 1 June 2007; revised 24 June 2008; accepted 12 Oct. 2008; published online 10 Dec. 2008.
For information on obtaining reprints of this article, please send E-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2007-06-0064. Digital Object Identifier no. 10.1109/TCBB.2008.133.

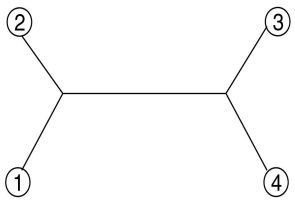

Fig. 1. A quartet tree is the smallest informative unrooted tree. The quartet tree $((1,2),(3,4))$.

Most quartet-based methods (e.g., [32], [26], [34]) first infer quartet trees from the aligned sequences normally by the maximum likelihood criterion and then use a combinatorial algorithm to solve the maximum quartet consistency problem, i.e., construct a tree satisfying the greatest number of these quartet trees (see exact definition in the Definitions part).

While the first task was heavily studied, including recent analytic algebraic solutions [9], [11], [10], [8], [7], the problem of devising good and fast heuristics for the second task has not been extensively pursued. Current methods (see, e.g., [28], [13], [32], [26]) handle conflicting inputs by ignoring the information causing these conflicts even when the degree of conflict is considerably low.

In this paper, we devise a new heuristic method for the second part of this task: The amalgamation of the quartet trees into a complete tree. Our method handles conflicting quartet trees naturally by recursively seeking a split that will violate only a small fraction of the quartets. We extend the divide and conquer recursive algorithm of Aho et al. [1] to the unrooted setting by splitting the set of taxa recursively. This is done by attempting to solve a MaxCut problem in a graph representing the quartets' relationships, using an extremely fast semidefinite programming (SDP)-like heuristic algorithm. Therefore, our method is based on a two-level heuristic approach: In the higher level, the quartet consistency problem is solved by the recursive, divide, and conquer high-level algorithm (described in Section 2.5), and in the lower level a heuristic SDP-like engine is employed for solving a MaxCut instance at every recursive stage of the algorithm (described in Section 2.5). Upon returning from the recursion, the two subsolutions need to be merged into a full tree over the whole taxa set while satisfying the greatest number of quartets. This raises a technical challenge that we solve.

A cardinal requirement from quartet-based methods is the ability to consider the confidence of each quartet [26]. Our algorithm enables this requirement very naturally by the weight of the edges in the graph corresponding to the quartet.

SDP in the use of phylogenetic reconstruction was employed previously by Ben-Dor et al. [4] for the same task as addressed here. They formulated the whole quartet problem as a SDP problem, solved it by some available SDP solver yielding an embedding in a space, computed the induced metric over the points and used a distance-based reconstruction method to build the tree. However, as their high-level algorithm used an exact, computationally intensive SDP algorithm, they were able to construct trees over $n = 15$ taxa at most. In contrast, our approach is based on the following ideas:

1. Instead of embedding the taxa into n-dimensional space we use a very efficient algorithm to embed the points in a three-dimensional sphere.
2. We compute a cut based on this embedding and solve the problem recursively on each subproblem (in comparison to the "cherry picking" type of [4]).
3. While [4] uses a constant ratio to penalize "bad pairing" with respect to "good pairing," we employ a binary parametric search in order to find the ratio that maximizes the value of the cut. The parameter may vary as does the embedding while our recursion proceeds.

Using the ideas above, and a new high-level algorithm that enables a recursive procedure while keeping track of the branching process of the subproblems generated, we were able to increase the size of problems handled (taxa and number of quartets) by a factor of hundreds over existing methods. For example, for a set of 100,000 quartets taken from a model tree over 10,000 taxa with 90 percent of the quartets correct, our algorithm reconstructed a tree satisfying 89 percent of the quartets in 90 minutes (see details in Section 3). We remark here that as this new method is based on two heuristic solutions that are interwoven into one another, it provides no guarantee, neither on its performance nor on its running time.

Additionally, our findings show that taking quartets spanning smaller portions (subtrees) of the target tree into the input set yields trees with higher similarity to the model tree over taking the quartets uniformly. This finding has also some support in our result on real data.

A quartet algorithm can serve also for supertree reconstruction. In this setting, the input is a set of small subtrees over an overlapping large set of taxa. The goal is to find a tree over the complete set of taxa such that some objective function is maximized. The adaptation of a quartet method to a supertree method requires some consideration such as split overrepresentation, locality of "good" cuts and alike. Addressing all these, we compared our technique with the popular well-regarded supertree method Matrix Representation with Parsimony (MRP) on five real data sets of input subtrees. In general, our method exhibits advantage over MRP in the Maximum Agreement Subtree (MAST) criterion but not in the Robinson-Foulds (RF). On input trees taken from the RBCL gene in which the diameter (WRT evolutionary distance) is relatively small, our method strictly outperforms MRP in both criteria.

2 Methods

In this section, we describe our algorithm, quartet MaxCut, QMC. Before we present the algorithm, we give an intuition to our approach of how to satisfy the maximum number of quartets. Next, we detail on the parametric search we employ in order to find the parameter maximizing the ratio between satisfied to violated quartets. Eventually, we present the complete algorithm. We remark that we cannot guarantee running time of the algorithm or the fraction of satisfied quartets since its core is the heuristic MaxCut which we have no bounds on its performance or time complexity.

2.1 Preliminaries

2.1.1 Graphs and Cuts

A graph G is defined as a pair (V, E), where V is a set of nodes, and E is a set of edges between the nodes $E = \{(u, v)|u, v \in V\}$. A weighted graph is a triple $G = (V, E, w)$ where w is a function that assigns weights to every edge in

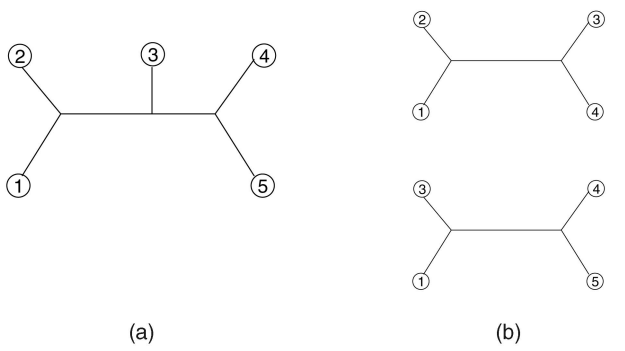

Fig. 2. (a) A phylogenetic tree over five taxa. (b) Two trees over four taxa induced by the tree on the left.

the graph. A tree is a connected graph with no cycles. A node in a tree is called a *leaf* if it is connected to a single other node only. Otherwise, it is *internal*. A *cut* $C = (S, \bar{S})$ in a graph is a partition over the nodes V (i.e., $S \cup \bar{S} = V$ and $S \cap \bar{S} = \emptyset$). An edge (u, v) is in the cut if $u \in S \wedge v \in \bar{S}$. We denote this set of edges by E_C. The *weight* of a cut C, $w(C)$ is the sum of weights of the edges in $C : \sum_{e \in E_C} w(e)$. A maximum cut $C^m = (S^m, \bar{S}^m)$ is a cut with maximum weight in G. The task of finding C^m is called the *MaxCut problem*.

2.1.2 Phylogenetic Trees

For a tree $T = (V, E)$, we denote by $\mathcal{L}(T)$ the set of leaves of T. A phylogenetic tree T over a set of taxa $\mathcal{X}$ is a tree for which there is a bijection between $\mathcal{X}$ and $\mathcal{L}(T)$. The removal of an edge e in a tree splits the tree into two subtrees, and therefore induces a split among the leaves of the tree. We identify an edge e by the split $(U, \mathcal{X} \setminus U)$ it generates and denote it by e_U (when U is arbitrarily one of the parts). Let T be a tree and $A \subseteq \mathcal{L}(T)$ a subset of the leaves of T. We denote by $T|_A$, the tree induced by A where all internal vertices with degree two contracted. For two trees T and T', we say that T *satisfies* T', and T' is *satisfied* by T, if $\mathcal{L}(T') \subseteq \mathcal{L}(T)$ and $T|_{\mathcal{L}(T')} = T'$. Otherwise, T' is *violated* by T. For a set of trees $\mathcal{T} = \{T_1, \ldots, T_k\}$ with possibly overlapping leaves, we say that $\mathcal{T}$ is *consistent* if there exists a tree T^* over the set of leaves $\bigcup_i \mathcal{L}(T_i)$ that satisfies every tree $T_i \in \mathcal{T}$ (see Fig. 2). Otherwise, $\mathcal{T}$ is *inconsistent*. When $\mathcal{T}$ is inconsistent, it is desirable to find a tree T^* over $\bigcup_i \mathcal{L}(T_i)$ that minimizes some objective function. T^* is denoted a *supertree* and the problem of finding T^* is the *supertree problem*.

A *quartet* tree (or just a quartet for short), is a tree over four leaves $\{a, b, c, d\}$. We write a quartet over $\{a, b, c, d\}$ as $((a, b), (c, d))$ if there exists an edge e_U such that $a, b \in U$ and $c, d \notin U$. When the input $\mathcal{T}$ to the supertree problem is a set of quartets (henceforth denoted $\mathcal{Q}$), this special case is denoted the *maximum quartet consistency* problem.

2.2 The MaxCut Intuition

A divide and conquer algorithm operates on the set of taxa $\mathcal{X}$ by first partitioning the set into a partition $\mathcal{P}$ of two or more parts. Then, it solves recursively the subproblems induced by each part, and merges the subsolutions obtained into a complete solution. In our case, the partition $\mathcal{P}$ is always of two parts denoted $\mathcal{P} = (\mathcal{X}', \bar{\mathcal{X}}')$.

Let $\mathcal{Q}$ be a set of quartets. A quartet $q = ((a, b), (c, d)) \in \mathcal{Q}$ is said to be *unaffected* by a partition $\mathcal{P}$, if all $\{a, b, c, d\}$ are in one part of $\mathcal{P}$. Otherwise, that is, not all $\{a, b, c, d\}$ are in one part of $\mathcal{P}$, it is *affected* by $\mathcal{P}$. For an affected $q = ((a, b), (c, d))$ we say that q is *satisfied by* $\mathcal{P}$ if exactly a and b are in $\mathcal{X}'$ and is *violated by* $\mathcal{P}$ if exactly a and c are in $\mathcal{X}'$ or exactly a and d are in $\mathcal{X}'$. Otherwise, we have $|\{a, b, c, d\} \cap \mathcal{X}'| = 3$ and we say that q is *deferred*. At every step in the recursion, some quartets are satisfied, some are violated and some continue to the next steps in the recursion (i.e., either deferred or unaffected). A plausible strategy is to maximize the ratio between satisfied and violated quartets at every step.

Given the set of quartets $\mathcal{Q}$ over the taxa set $\mathcal{X}$, we build the following weighted (multi) graph $G = G(\mathcal{Q}) = (V, E)$ with $V = \mathcal{X}$ and E as follows: For every $q = ((a, b), (c, d)) \in \mathcal{Q}$ we add the edges $\{(a, c), (a, d), (b, c), (b, d), (a, b), (c, d)\}$ to E. We distinguish between the edges and denote the edges $\{(a, c), (a, d), (b, c), (b, d)\}$ as *good edges* and $\{(a, b), (c, d)\}$ as *bad edges* (see Fig. 3). Observe that between two nodes in $G(\mathcal{Q})$ there can be good and bad edges simultaneously, originated from different quartets. For example, in Fig. 4 there are one good edge and one bad edge between nodes 1 and 3 originated from quartets $((1, 2), (3, 4))$ and $((1, 3), (4, 5))$, respectively. We denote by E_g and E_b, the set of good and bad edges, respectively, and observe that $E \setminus E_g = E_b$.

We note that a cut in G corresponds to a partition over the set of taxa. Given a cut $C = (S, \bar{S})$ in the graph corresponding to a partition $\mathcal{P}$, we notice:

Observation 1. *For an affected quartet $q \in \mathcal{Q}$*

1. *q contributes four good edges to the cut if q is satisfied.*
2. *q contributes two good edges and one bad edge to the cut if q is deferred.*
3. *q contributes two good edges and two bad edges to the cut if q is violated.*

Fig. 4 shows graphically the effect of a cut in a graph on the two quartets generating that graph. The above observation

Fig. 3. For a quartet tree $((a, b), (c, d))$ we draw (a) the four good edges (solid blue) $\{(a, c), (a, d), (b, c), (b, d)\}$ and (b) the two bad edges (dashed red) $\{(a, b), (c, d)\}$.

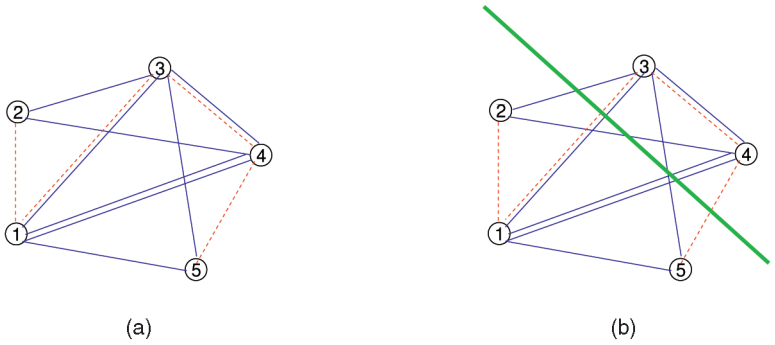

Fig. 4. (a) The graph induced by the quartets $((1,2),(3,4))$ and $((1,3),(4,5))$ from Fig. 2. (b) A cut separating $\{1,2,5\}$ from $\{3,4\}$, therefore, satisfies quartet $((1,2),(3,4))$ but violates $((1,3),(4,5))$, and hence contains six good and two bad edges.

links between the number of quartets satisfied/violated/deferred and the number of good/bad edges in the cut.

2.3 The Heuristic SDP-Like MaxCut

In this section, we detail on our heuristic for finding a good cut in the graph representing the set of input quartet trees. The final stage of this heuristic entails solving a MaxCut problem. For this restricted task, we used the SDP-like MaxCut heuristic (the lower level heuristic) we devised in [29] for a related but rather different problem of rooted trees consistency.

Our algorithm is based on maximizing the ratio between satisfied and violated quartets. By the discussion in the previous section, one approach can aim at finding a cut which maximizes the ratio between good and bad edges, at every instance of the recursive algorithm. A weaker variant aims at finding a cut C maximizing the function

$$\left(\sum_{e \text{ is a good edge}} \delta_{C,e}\right) - \left(\alpha \sum_{e \text{ is a bad edge}} \delta_{C,e}\right), \qquad (1)$$

where $\alpha > 0$ is a *weight ratio parameter* between the good edges to bad edges in C and

$$\delta_{C,e} = \begin{cases} 1, & \text{if } e \in C, \\ 0, & \text{otherwise.} \end{cases}$$

Solving (1) is equivalent to finding a maximum cut (solving MaxCut) in a graph in which good edges have unitary weight and bad edges, weight $-\alpha$. Unfortunately, in general, finding such a cut is NP-complete [17] as well as the original problem (satisfying maximum quartets). Several SDP-based approximation algorithms, however, have been suggested for related problems [18], [2]. These algorithms give a guarantee on the deviation of their solution from the optimal solution and work in polynomial time. The principle is to embed the vertices on the unit n-dimensional sphere (recall, $\mathrm{n} = \#\mathrm{taxa}$) so as to maximize the function:

$$\left(\sum_{\text{good edges } e=(i,j)} d(i,j)\right) - \left(\alpha \sum_{\text{bad edges } e=(i,j)} d(i,j)\right), \qquad (2)$$

where $d(i,j)$ is the euclidean distance between points i and j. Once the embedding is obtained, a hyperplane through the origin is selected randomly with the intuition that points embedded far apart from each other are likely to be in different sides of the hyperplane versus close by points. For example, Fig. 5b corresponds to a possible embedding of the graph in Fig. 5a. We see that points connected with good edges are located far apart on the sphere versus points connected by bad edges.

While the embedding stage can be handled by an accurate, computationally intensive SDP algorithm that guarantees the required separation for expected approximation ratio [18], [2], in [29], in the context of optimizing rooted trees consistency, we found that a random embedding of the points on a three-dimensional sphere (versus n-dimensional) and subsequently moving the points locally in order to optimize (2) suffices: it is extremely fast and

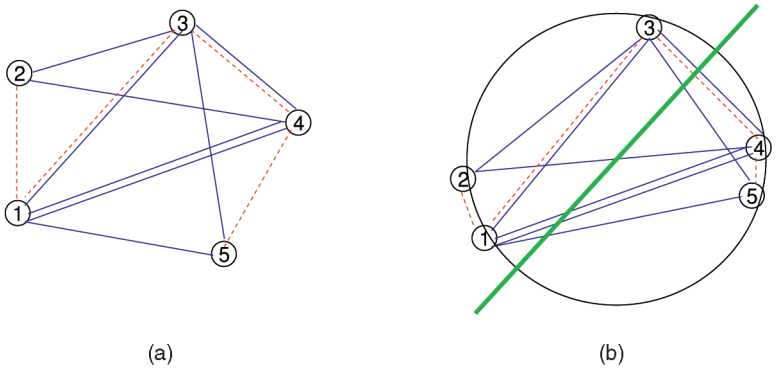

Fig. 5. (a) The graph corresponds to the set of quartets $((1,2),(3,4))$ and $((1,3),(4,5))$ (see Fig. 4). (b) The embedding is a possible embedding according to (2). Note that points 1 and 2 are positioned nearby as well as points 4 and 5 as they are separated by bad edges only. As can be seen, the hyperplane separates $\{1,2,3\}$ from $\{4,5\}$ which is indeed a split in the tree induced by these two quartet trees (see Fig. 2).

produces results accurate enough for our needs. This heuristic SDP-like engine is the explanation for the very small running times of our algorithms. The Heuristic MaxCut algorithm appears in Appendix A, and fuller details of it appear in [29].

2.4 Dynamic Ratio Parameter

In [29], we used a constant weight ratio parameter α in (2) (specifically $\alpha = 3$). While for the application of [29] (amalgamating rooted triplets) this constant ratio sufficed, in the current setting, however, working with a constant α did not yield good enough results. Moreover, for some combinations of input quartets and α, there is no cut with positive weight and the *empty cut* $C_\emptyset = (V, \emptyset)$ is the maximum cut (we illustrate this by an example in the following). Returning an empty cut from the MaxCut procedure is prohibitive in that type of a divide and conquer algorithm, as in order for the algorithm to proceed, a nonempty cut must be generated at every step.

In this paper, we employed the technique of *parametric search* for finding the optimal weight ratio parameter α. We now elaborate on this technique.

Definition 1. *For a cut $C = (S, \bar{S})$, let the* edge ratio *of C, $\rho(C)$, be the ratio between the number of good and bad edges in C.*

We denote by $C^* = (S^*, \bar{S^*})$ the cut maximizing ρ and let $\rho^* = \rho(C^*)$ be the optimal edge ratio in G. Recall that we are interested in finding ρ^* at every recursion step. Note that C^* (and hence also ρ^*) is a combinatorial property of G independent of any edge weight function w (in contrast to the maximum cut $C^m = (S^m, \overline{S^m})$ that *is* dependent on a particular w). However, as our MaxCut algorithm finds a maximum cut in a graph, we will show that selecting the weight ratio parameter α more wisely, can lead to a bigger ratio ρ.

We illustrate the influence of α on ρ^* by an example: Consider a graph with one cut C' with 200 good edges and 10 bad ones, and another cut C'' with 30 good edges and a single bad edge. Clearly C'' is better with $\rho(C'') = 30 > \rho(C') = 20$. Now, for $\alpha = 10$, we have by (1) $w(C') = 200 - \alpha \cdot 10 = 100$ versus $w(C'') = 30 - \alpha \cdot 1 = 20$. In contrast, for $\alpha = 100$, we have $w(C'') = -800$ versus $w(C'') = -70$ and both are inferior to $w(C_\emptyset) = 0$. The optimal α is $30 - \varepsilon$ for some small ε for which $w(C'')$ is bigger than both $w(C')$ and $w(C_\emptyset) = 0$.

The example above shows that in order to find ρ^*, the optimal α should be sought. The following observation shows how α should be set such that a nonempty cut is returned (i.e., $C_\emptyset$ is not the maximum cut):

Observation 2. *Let $\alpha > 0$ be the weight ratio parameter and assume there is a cut C' with $\rho(C') > \alpha$. Then, the maximum cut (in the weighted graph G) C^m satisfies $w(C^m) > w(C_\emptyset)$.*

Proof. Obviously, the empty cut $C_\emptyset$ satisfies $w(C_\emptyset) = 0$. However, for C', by (1) we have

$$\begin{aligned} w(C') &= |E_g \cup E_{c'}| - \alpha |E_b \cap E_{C'}| \\ &\quad \text{(by the definition of } \rho) \\ &= (\rho(C') - \alpha)|E_b \cap E_{C'}| \\ &\quad \text{(by the assumption)} \\ &> 0. \end{aligned}$$

Therefore, we obtain:

$$w(C^m) \geq w(C') > 0 = w(C_\emptyset).$$

□

Similarly, we observe

Observation 3. *For a cut $C' = (S', \bar{S'})$ with $\rho(C') < \alpha$, $w(C') < w(C_\emptyset)$.*

Corollary 1. *C^* (the cut maximizing ρ) is the maximum cut obtained by the maximum α s.t. $\alpha < \rho^*$.*

Proof. By Observation 2, we obtain that $w(C^*) > w(C_\emptyset) = 0$ and since $\rho(C^*) > \rho(C')$ for every cut $C' \neq C^*$, by Observation 3 we obtain $w(C') < w(C_\emptyset)$. □

Our Parametric MaxCut algorithm searches for the cut C^*. By Corollary 1, in order to find C^* we need to find the biggest α s.t. a nonempty cut is returned. In order to bound the search space we need to find upper and lower bounds for ρ^* (and by Observations 3 and 2, also for α). The following observations establish these bounds. We first define two special cuts in a graph.

Definition 2. *For a vertex $v \in V$, the cut $(V \setminus v, \{v\})$ is called a* singleton cut. *We also say that a cut is* perfect *if there are no bad edges in it.*

Observation 4. *Let $G = G(\mathcal{Q})$, Then, for any singleton cut C_s in G, $\rho(C_s) = 2$.*

For example, in Fig. 4, the cut $(\{1\}, V \setminus \{1\})$ has two bad and four good edges, the cut $(\{2\}, V \setminus \{2\})$ has one bad and two good edges, and so on.

Proof. Let $C_s = (V \setminus v, \{v\})$. Then, the only quartets affected are of the type $((a, b), (c, v))$. The observation follows by the fact that each such quartet adds to C_s the two good edges (a, v) and (b, v) and the bad edge (c, v). □

Corollary 2. $\rho^* \geq 2$.

Proof. Note that a graph with $n = 2$ contains one singleton cut and for $n > 2$ there are exactly n singleton cuts. By Observation 4 the claim follows. □

By the discussion above, in order to find ρ^*, we need to increase α until we hit an empty cut. There are, however, some complications.

Observation 5. *Let $G = G(\mathcal{Q})$ and E_g the set of good edges. Then, ρ^* is defined and is at most $|E_g| = 4|\mathcal{Q}|$ iff G has no perfect cut.*

Proof. $\Longrightarrow$ There are exactly $4|\mathcal{Q}|$ good edges in the graph. Since there is no perfect cut, any cut contains at least a single bad edge and we get $\rho^* \leq 4|\mathcal{Q}|$.

$\Longleftarrow$ There is a perfect cut C^p, that is $E_{C^p} \cap E_b = \emptyset$. By the definition of ρ, $\rho(C^p)$ is undefined. □

Finally, since we are searching for ρ^* by finding a maximum cut in the weighted graph where the weights are determined by the edge weight parameter α, we can use the observations above to set a limit for α.

Observation 6. *If $\alpha > 4|\mathcal{Q}|$ and a nonempty cut is returned, then G has a perfect cut.*

Proof. By Observation 2, if there is a cut C' with edge ratio $\rho(C') > \alpha$ then the maximum cut in the graph is not the empty cut $C_\emptyset$. Conversely, by Observation 5 there is no cut with $\rho > 4|\mathcal{Q}|$, we obtain that the cut returned is a perfect cut. □

The above observations lead to a binary search algorithm for finding the cut C^* achieving ρ^*. In the first phase, by Observations 6 and 4 we find the space boundaries and at the second phase, by Observations 2 and 3 we perform the binary search in that space. The code is below:

Parametric MaxCut $(G(\mathcal{Q}))$

1. $\alpha \leftarrow 2$
2. while $(\alpha \leq |E_g|)$
   (a) for every $e \in E$, $w(e) \leftarrow \begin{cases} -\alpha & e \in E_b \\ 1 & \text{otherwise} \end{cases}$
   (b) $C \leftarrow MaxCut(G, w)$
   (c) if C is the empty cut, go to 3 (α is too high).
   (d) $\alpha \leftarrow 2\alpha$
3. if $\alpha > |E_g|$ return the perfect cut found.
4. $b_u \leftarrow \alpha$
5. $b_l \leftarrow \frac{\alpha}{2}$
6. while $(b_u - b_l > \varepsilon)$
   (a) $\alpha \leftarrow \frac{b_u + b_l}{2}$
   (b) for every $e \in E$, $w(e) \leftarrow \begin{cases} -\alpha & e \in E_b \\ 1 & \text{otherwise} \end{cases}$
   (c) $C \leftarrow MaxCut(G, w)$
   (d) if C is the empty cut,
      - $b_u \leftarrow \alpha$
   (e) else
      - $b_l \leftarrow \alpha$

In step 2 in the algorithm above, we find the upper and lower bounds for the optimal α while in step 6 we perform a binary search for the best α in that interval. Since we end this search when the interval is of size at most ε and there is a nonempty cut in the interval, we obtain the following observation:

Observation 7. *If MaxCut at steps 2b and 6c returns the maximum cut in G, the algorithm above returns a cut C with* $|\rho(C) - \rho^*| < \varepsilon$.

In addition, since the search space is decreasing by half each time, we get:

Corollary 3. *If the MaxCut at steps 2b and 6c returns the maximum cut in G, then it is invoked* $O(\log|\mathcal{Q}|)$ *times for every search.*

As was indicated above, in practice, we employed our heuristic MaxCut algorithm in steps 2b and 6c in Parametric MaxCut

2.5 The High-Level Algorithm

A big difficulty when moving from a rooted setting to an unrooted one, is that the relationship between the subtrees is not embodied by the least common ancestor property, rather by tree edges that induce splits (or cuts) on the taxa set (see Fig. 6 for a specific example) . The generic algorithm to handle subtrees in a rooted setting operates on rooted triplets. It solves the problem when all the triples are consistent and was developed in the context of relational databases by Aho et al. [1]. Steel presented the algorithm for use in phylogenetics in [31]. For the sake of completeness, we give the formal algorithm in Appendix. In this naive recursive type algorithm, every subtree rooted at some ancestor v is linked to the ancestor generated one level above it in the recursion procedure. The invariant is that if the triplets are consistent, we are guaranteed to generate the common ancestors at the correct order. In contrast, here we cannot expect to obtain the cuts at the appropriate order. A specific example is depicted in Fig. 6. The tree over six leaves at the left (Fig. 6a) is represented by three quartets in the middle. A naive recursive implementation of the type of Aho et al. acting in the following order (from left to right) of splits: $(\{1,2\},\{3,4,5,6\})$, $(\{3,4\},\{5,6\})$ will produce the erroneous tree in Fig. 6b. Note that both splits are correct and are supported by the set of quartets.

Another complication with respect to the rooted setting is that in a recursive rooted setting, at every recursive step, every triplet is either satisfied, violated, or wholly contained in one of the parts. Dealing with quartets, the above is not true: A quartet can be deferred to successive steps, yet not being wholly contained in any of the parts. The challenge is to keep track of these quartets and try to satisfy them.

In order to cope with the above hurdles, we devised the following algorithm: At every step of the algorithm, we compute a parametric MaxCut as defined above. We now solve each part recursively but with the following two additions. We augment each part with a new artificial taxa x_1 (or x_2, respectively) and replace the single taxon in every deferred quartet that is in the counter part with that new taxon. Upon returning from the recursions on both parts, we join the two trees by removing the two new taxa and joining their corresponding pendant edges as a single edge. Fig. 7 illustrates schematically how the two parts $\{1,2,3\}$ and $\{4,5,6\}$ were augmented with artificial taxa x_1 (or x_2, respectively). On return from the recursion, after the two subtrees were inferred at successive stages, these two species are removed to form the tree in Fig. 7b.

We illustrate how the algorithm handles the same set of quartets and the same problematic order of splits. Fig. 8 shows the recursive invocation tree on that input. In Fig. 8a, every node signifies invocation of the Parametric MaxCut algorithm with the set of input quartets. Along the down-arrows appear the taxa passing to the subsequent step

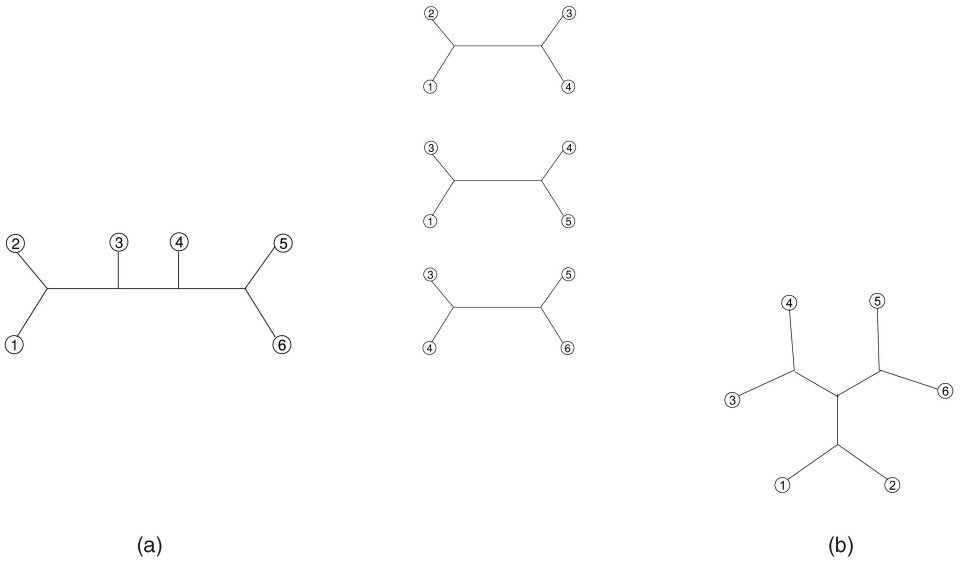

Fig. 6. Erroneous inference of the splits by a naive divide and conquer approach. The erroneous split $(\{3,4\},\{1,2,5,6\})$ at the resulted inferred tree in (b) is not a split of the original tree in (a).

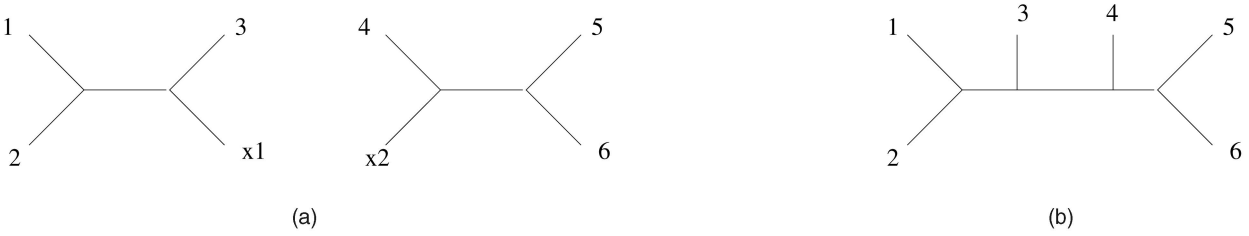

Fig. 7. (a) The two subtrees are joined by removing the two artificial nodes and connecting their corresponding (b) two pendant edges.

(including the added artificial taxon). In Fig. 8b, the recursion rollback is depicted with the returned subtrees along the edges. It can be seen how at every node, the artificial taxon added is removed and its pendant edges are joined, up to the point where the correct tree is constructed (top edge on right tree, Fig. 8b).

The formal algorithm is depicted below (for sake of clarity, we denote the new taxa by x_1 and x_2 although they need to be uniquely identified across all levels[1]).

QMC $(V, \mathcal{Q})$

1. Let V be the set of taxa.
2. If $\mathcal{Q} = \emptyset$ return a tree of depth 1 with all V sister taxa.
3. $C = (V_1, V_2) \leftarrow$ Parametric MaxCut$(G(\mathcal{Q}))$
4. for $i = 1, 2$
   - $V_i \leftarrow V_i \cup \{x_i\}$ where x_i is a new artificial taxon
   - $\mathcal{Q}_i \leftarrow \{q = ((a,b),(c,d)) \in \mathcal{Q} : a,b,c,d \in V_i\}$
   - for every $q = ((a,b),(c,d)) \in \mathcal{Q} : a,b,c \in V_i \wedge d \in V_{3-i}$, $\mathcal{Q}_i \leftarrow q = ((a,b),(c,x_i))$
   - $T_i \leftarrow$ QMC$(V_i, \mathcal{Q}_i)$
5. join T_1 and T_2 by removing x_1, x_2 and connecting their corresponding two pendant edges.
6. return the resulted tree

At every recursive step, the algorithm above adds a new taxon. Hence, if a singleton cut is returned, the algorithm might run forever. However, as by Observation 4 a singleton cut C_s has ratio $\rho(C_s) = 2$, it can be shown that there are always nonsingleton cuts with ratio at least 2. Assuming we can find these cuts (by the properties of Parametric MaxCut), the algorithm will terminate.

The above arguments relied on the assumption of finding a maximum cut in the graph. The following claim relies on a stronger assumption that we can find all the splits of the inputs:

Claim. Suppose the set of quartets is consistent with some tree T, and the algorithm *QMC* finds at every step, a split consistent with the input set of quartet (for that step). Then, it returns a tree consistent with the primal input of quartets.

Proof. We prove the claim by induction on the number of splits k in the tree. For $k = 1$, the algorithm finds the single split and no quartet passes to further step. The algorithm will return a tree with two internal nodes and all taxa at one side connected as sister taxa to each of these nodes. Assume that the claim holds for $k = n$ and we prove for $k = n + 1$. Then, the algorithm finds a correct split and divides the taxa set to two subsets. By the induction assumption, for each of the subsets the algorithm finds a tree in which every quartet wholly contained in the subset is satisfied. For a deferred quartet, the new quartet with the artificial taxon is also satisfied and by the way of joining the subtrees, the original quartet is satisfied as well. □

2.6 Short Quartets Difficulties

A quartet $q = ((a,b),(c,d))$ is said to *define* an edge $e = (u,v)$ in T, if upon removal of u, v from T, each of a, b, c, d resides in a different subtree of the four subtrees generated. We root

1. This can be achieved by concatenating the level of the recursion to the names x_1 and x_2.

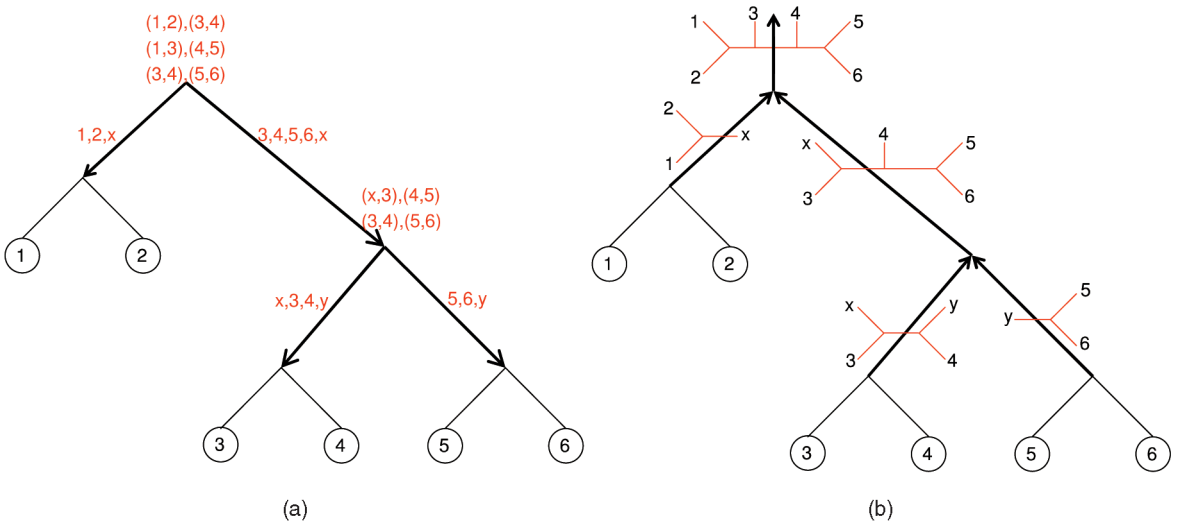

Fig. 8. The operation of the high-level QMC algorithm on the quartets $\{((1,2),(3,4)),((1,3),(4,5)),((3,4,),(5,6))\}$. (a) The recursive calls with the input quartets at every call (node) and the splitted taxa along the edges. (b) The rollback from the recursion with the returned trees along the edges.

each subtree at the internal node neighboring either u or v. A $q=((a,b),(c,d))$ defining an edge $e=(u,v)$ is *short for* e if each of a,b,c,d is closest to the root in its subtree[2] (see Fig. 9).

Short quartets are a suitable input for tree reconstruction [14], [15], [13], [19]. Moreover, when the set of quartets $\mathcal{Q}$ is consistent with some tree T and all short quartets of T are in $\mathcal{Q}$, there exists a polynomial time algorithm that finds T in $O(n^5)$ [14]. However, short quartets pose an obstacle on a divide and conquer approach such as described here. As short quartets are *local* in the sense that each such quartet spans a small part of the tree, the maximum cut induced by quartets, contains several splits of the tree T as illustrated by the example in Fig. 10. The tree in Fig. 10a gives rise to the three short quartets in Fig. 10b. Now, every edge in the tree satisfies one quartet and defers another. Suppose the difference (in gain) between a satisfied and a deferred quartet is ε. Then, the cut $\{\{1,2,5,6\}\{3,4\}\}$ gives a gain of 2ε versus ε by any real split of the tree.

Fig. 11 illustrates the same effect on bigger, more realistic inputs. Clearly, these cuts (as in Figs. 10 and 11) are not real splits in the tree and a naive application of the *Parametric MaxCut* algorithm will return the erroneous splits as illustrated in the figures.

2.6.1 Connected Components Enhancements

In order to cope with such inputs, we devised the following heuristic that incorporates information from the cut found and the information from the quartets. We first run the *Parametric MaxCut* algorithm and keep the returned split aside. The idea is that indeed on a global view, the cut depicted in Fig. 11b maximizes ρ; however, it also incurs many violated quartets. We wish to minimize these violations *between* parts on account of violations *within* parts. We start by handling "safe parings"—taxa that are cherries (sister taxa) in quartets, and join them. We now merge parts maximizing violations *between* themselves. This is, similarly to the short quartets approach, a local view, rather than a global view, as represented by the maximum cut approach. We believe this is the reason that the heuristic works on such inputs.

We define a partition $\{S_i\}$ on the taxa where in the initial partition, every taxon is a singleton. We now merge two subsets S_{i_1},S_{i_2} if there are two taxa i_1,i_2 such that $i_1\in S_{i_1}$ and $i_2\in S_{i_2}$, i_1,i_2 are in the same part of the cut returned from MaxCut, and there is a quartet $((i_1,i_2)(x_1,x_2))$ for any taxa x_1,x_2. A *violation* between two subsets is a quartet violated by the subsets (e.g., $q=((a,b),(c,d))$ and a,c in one set while b,d in the other). Given the set of subsets obtained from the first stage, we now merge iteratively the two components with the maximum number of violations. The merging is terminated when the number of components is two. Although we do not have formal proof for the performance of this enhancement, we tried it on some of our data prone to such a phenomenon (geometrical distribution and short quartets; see experiments in Section 3) and it led to improvement of up to 5 percent in quartet score and even in the other scores.

3 Results

In order to test our method, we conducted experiments that compared our method versus the supertree method MRP.

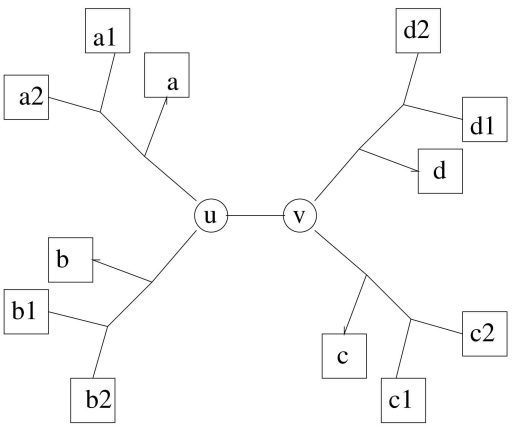

Fig. 9. The quartet $q=((a,b),(c,d))$ defines $e=(u,v)$ and is short for e.

2. This is a slightly different definition than [14], [19].

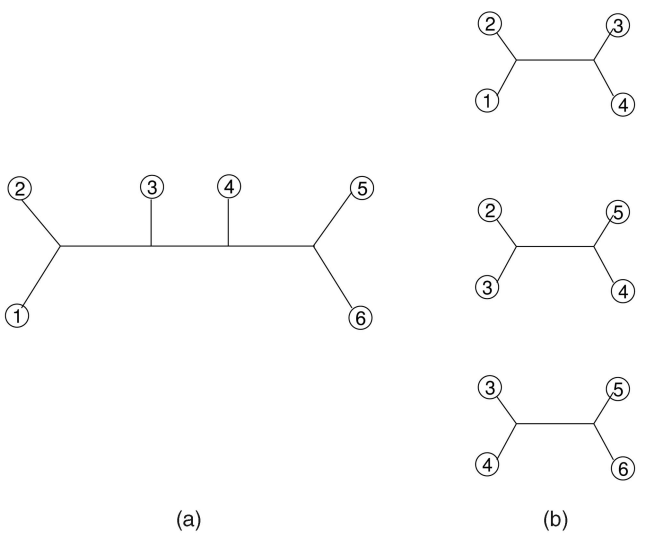

Fig. 10. (a) A tree over six leaves and (b) the three short quartets it induces. A MaxCut will put {3,4} in one part and {1,2,5,6} in another as it yields a better cut than any cut induced by a tree edge.

MRP [3], [24] is the most widely used supertree method by practitioners and it has been found to have good performance [16]. In this method, the input subtrees are encoded in a $\{0,1\}$ matrix in the following way: As every edge e in an input subtree T_i induces a partition on $\mathcal{L}(T_i)$, (A,B), e is encoded as binary character C where for each $s \in A$, $C(s) = 0$ and for $s \in B$, $C(s) = 1$. For $s \in \mathcal{X} \setminus \{A \cup B\}$, $C(s) = ?$ indicating a *missing state*. The method tries to find the maximum parsimonious tree w.r.t. that matrix.

Observation 8. *Let T be a tree over $\mathcal{X}$ and M an $n \times m$ matrix as defined above. Then, for every character C in M, the parsimony score of C w.r.t. T is 1 if the quartet coded for C is satisfied by T. Otherwise it is 2.*

The above observation implies that on a quartet input, the objective function that MRP tries to optimize, the MRP score, is exactly the maximum quartet consistency score. Since MRP uses state-of-the-art techniques to solve the problem, this makes it competitive to our method. We remark that MRP is *not* a quartet method and can be used in wide range of tasks. However, as shown above, it can be specialized to our goal as well.

Our MRP implementation used heuristic searches in PAUP* for maximum parsimony saving the 200 best trees in memory. From previous experience [30] and experimentation, we set the number of random sequence additions to 3. Allowing a longer search (more iterations) had an extremely small impact on the topological accuracy of the resulting tree but increased linearly MRP's running time, making its computation impractical. Therefore, the parameters we used are apparently optimal for MRP. The PAUP* command we used for our heuristic search is as follows:

```
set Warnreset=no Autoclose=yes MaxTrees=200;
hsearch NReps=3 Addseq=random;
contree all/strict=no majrule=yes;
```

3.1 Simulated Data

In this type of experiments, the input to our method was a set of quartets generated from a set of model trees. For each number of taxa $n = 100, 200, 300$, we generated 10 (random, unweighted) trees and for each tree we generated three data sets, differing in their size (number of quartets $|\mathcal{Q}|$) with $|\mathcal{Q}| = n^{1.5}, n^{1.7}, n^2$. Of this set of quartets, a constant portion was flipped, i.e., was changed to disagree with the model tree. Specifically, we flipped either 10 or 30 percent of the quartets (yielding 90 and 70 percent correct quartets, respectively). The properties we wanted to measure are the accuracy of the tree returned in terms of the number of quartets satisfied, the resemblance (topological accuracy) of that tree to the model tree and the running times. As both methods satisfied approximately the same percentage of correct quartets, we do not report on this criterion rather only on topological accuracy and running time.

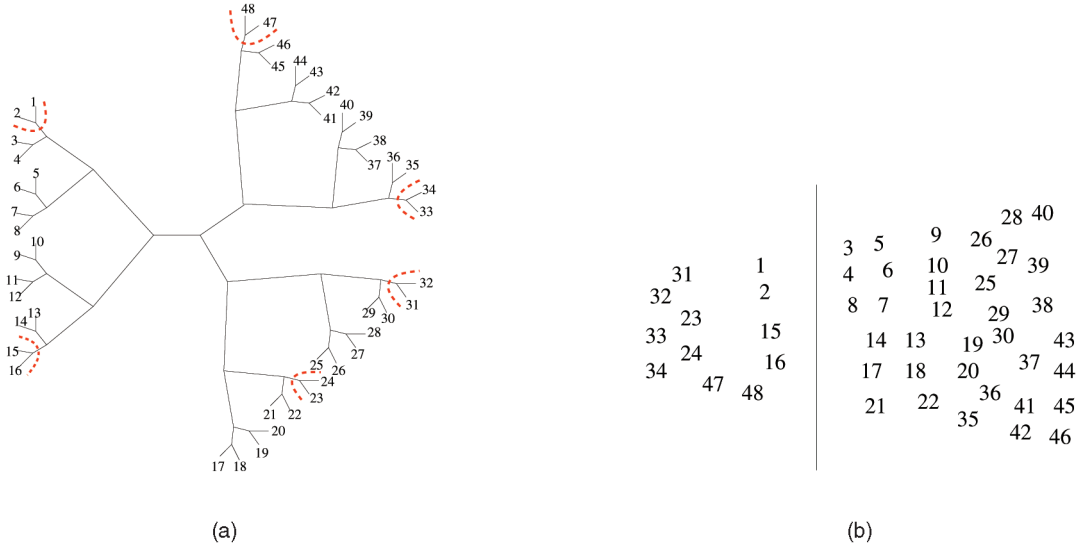

Fig. 11. A MaxCut on a local set of quartets, (a) will separate the cherries marked with a red dashed arcs from the rest of the subtree and (b) will locate them together in a single part.

TABLE 1
Results from Experiments of Uniform Distribution of Quartet Selection

#taxa	#quartets	MAST distance				%FP/FN				Running Time			
		MRP		QMC		MRP		QMC		MRP		QMC	
		70%	90%	70%	90%	70%	90%	70%	90%	70%	90%	70%	90%
100	1000	84	74	77	69	100/100	87/87	98/99	82/90	258	280	21	20
100	2,511	71	62	67	57	87/87	71/71	86/91	62/74	768	497	29	32
100	10,000	60	50	49	43	62/62	39/39	55/62	28/42	2657	2226	41	48
200	2,828	89	80	82	76	100/100	93/93	99/100	89/95	6066	4154	53	46
200	8,161	77	74	75	70	94/94	85/85	90/94	75/86	23173	23541	91	74
200	40,000	65	57	55	51	73/73	57/57	69/76	45/60	85163	50243	122	119
300	5,196	93	84	87	74	100/100	98/98	98/99	89/94	24412	23463	52	58
300	16,259	83	79	77	75	95/95	89/89	95/97	82/89	35477	32757	110	216
300	90,000	70	65	57	48	91/91	65/65	73/80	49/66	184207	163763	209	189

The left two columns represent the input type (from left): size of model tree (#taxa), and the number of input quartets ($|\mathcal{Q}| = n^{1.5}, n^{1.7}, n^2$). For each of the three criteria, results for both error rates (70 and 90 percent) are presented side by side. For every criteria we show the MRP results versus QMC. The result columns (right three meta columns) are: size (#species) of the biggest tree satisfied by both model and inferred trees (MAST), false positive/false negative, and running times.

The standard practice in the field to measure the topological accuracy is to report Robinson-Foulds symmetric difference distance; however, the use of RF distances has been rightly criticized [25] as too crude, and so we elected to report False Negative (FN) rates (also called Missing Edge rates) and False Positive (FP) rates. We note that the RF error rate is the average of these two error rates. We will comment on this in a more general comparison between the two methods. We also computed the MAST error rate, which is the percentage of the taxa that must be removed from both trees, in order to make the two trees identical. This error rate has been used in some studies evaluating phylogenetic accuracy (see, for example, [16]).

- *FP rate*: The splits (edges) in the estimated tree T which do not appear in the model tree T_0 are "false positives." This value is normalized by the number of internal edges in the estimated tree, to produce a value between 0 and 100 percent.
- *FN rate*: The splits in the model tree T_0 which do not appear in the estimated tree T are "false negatives." This value is normalized by the number of internal edges in the model tree to produce a value between 0 and 100 percent.
- *MAST distance*: Let T be a tree over a taxa set S and $A \subseteq S$. We denote by $T|_A$ the tree induced by the subset A of leaves, so that all degree-two nodes are suppressed. The MAST score between two trees T_1 and T_2 is the size of the largest subset S, such that $T_1|_S = T_2|_S$. To produce the MAST distance between T_1 and T_2, we first normalize the MAST score by the number of leaves common to T_1 and T_2, and then subtract the result from 100 percent. Note that if the MAST distance between two trees is 0, then the two trees are identical.

We conducted two types of simulations differing by the way the species of any quartet were chosen.

1. **Uniform.** The species are chosen according to a uniform distribution from the species set.
2. **Geometrical.** Quartets over species with diameter d (the maximum number of edges on the model tree between any of the $\binom{4}{2}$ pairs of species) were chosen with probability proportional to $\frac{1}{d}$. This approach favors quartets of closely related species (low diameter) over ones which are of distant species.

3.1.1 Uniform Distribution Results

The results of our experiments with uniform quartet selection are depicted in Table 1.

The numbers confirm the anticipated result that the bigger the number of correct quartets the higher the trees' similarity, and this holds for every size of tree. The same holds also when the percentage of correct quartets is fixed and the number of quartets grows.

We see that the similarity between the trees is very sensitive to either the number of quartets or the absence of noise (percent correct quartets) and a significant similarity is obtained only when both parameters are relatively high. Another interesting characteristic property that emerges is that the bigger the tree (n) the *relative* amount of quartets required to maintain the same topological accuracy increases. This means that when #quartets is $|\mathcal{Q}| = n^{1.7}$, for example, the bigger the n the lower the topological accuracy. This is apparent for every number of quartets ($|\mathcal{Q}| = n^{1.5}, n^{1.7}, n^2$).

As to the comparison between MRP and our method (QMC), QMC strictly outperforms MRP in the MAST distance and the FP rate. However, even if for some parameters MRP is better in the FN rate, for all parameter ranges, QMC is better in the average of FP and FN, the RF rate. A conspicuous property is that the advantage of QMC over MRP is at its peak when the signal in the data increases, i.e., when the results with both methods are better. Therefore, for all ns, the difference in performance (RF score) was highest at 90 percent correct quartets and $|\mathcal{Q}| = n^2$. This may indicate a better scalability of our method to higher accuracy. We don't know whether this is a result of greater capability to handle larger data or other factors. Most notable, of course, is the advantage in the

TABLE 2
Data from Experiments of Geometrical Distribution of Quartet Selection

#taxa	#quartets	MAST distance				%FP/FN				Running Time			
		MRP		QMC		MRP		QMC		MRP		QMC	
		70%	90%	70%	90%	70%	90%	70%	90%	70%	90%	70%	90%
100	1000	76	60	70	54	84/84	53/54	80/83	44/50	342	283	33	39
100	2,511	58	43	56	39	55/55	27/27	52/54	20/23	645	372	40	46
100	10,000	39	41	29	30	19/20	15/17	12/13	5/6	2130	1860	49	56
200	2,828	82	60	73	54	86/86	46/46	78/82	41/47	5851	4536	83	63
200	8,161	54	47	44	36	43/43	20/20	45/47	13/15	22627	12508	118	90
200	40,000	44	38	36	30	13/13	14/14	10/10	6/6	50695	33538	142	155
300	5,196	83	57	80	57	83/83	40/40	75/77	43/46	23949	19263	139	92
300	16,259	57	52	53	44	43/43	17/17	45/47	14/16	37790	33315	136	142
300	90,000	48	44	46	40	14/15	13/13	12/13	6/7	187177	168155	225	254

running time. For the larger inputs ($n = 300$ or large #quartets), this ratio exceeded 1,000 many times. Moreover, the running time of MRP became prohibitive far before the limits of our methods (data not shown).

The difference in speed is not only "bonus" of our method. It enables us to reach problem sizes that are beyond the horizon for current techniques. For example, an MRP run of 10,000 quartets on 800 taxa ran more than a week without terminating. In contrast, the largest instance we tried our method on was of 100,000 quartets drawn from a model tree over 10,000 taxa with precision of 90 percent. Our method reconstructed a tree in about 90 minutes that satisfied 89 percent of the quartets.[3]

3.1.2 Geometrical Distribution Results

In this type of experiment, we gave preferences to "close" over "far" quartets. That means that a quartet of diameter d was chosen with probability $\frac{1}{d}$. We denote that as the *geometrical distribution*. The biological reasoning for using that distribution is that, in general, we are less certain about the order of speciation of distantly related species, than of more closely related species [21], [20], [14], [13], [19], and therefore we "weigh" these distant quartets less. This approach was proved useful in our study of sequence-based reconstruction [30]. The experiments were performed identical to the uniform distribution in terms of the number of taxa, number of quartets, and percent correct. Table 2 depicts our results on this distribution (for explanation see Table 1).

Under this distribution as well, the quartet fit measure stays similar to the values obtained under the uniform distribution (and therefore not shown). As can be seen, in contrast to the uniform distribution, the measures for tree similarity (MAST and FP/FN) attain very high values (low error rate) even for 70 percent accuracy (see, e.g., QMC results for $n = 200$ and #quartets $= n^2$) or small #quartets $= n^{1.7}$ (e.g., QMC results for $n = 200$ and 90 percent accuracy). Another noteworthy property is that in contrast to the uniform distribution, here an increase in the #species n, does not require an increase in the relative number of quartets (i.e., similar results are obtained at #quartets $|\mathcal{Q}| = n^{1.7}$ for all ns). Here also the differences between the two accuracy values (90 and 70 percent) are very significant, at both methods (MRP and QMC) and at all parameter ranges. We do not explore this point and leave it as an open question for future research. The advantage of QMC over MRP is more significant here than under the uniform distribution. In particular, the phenomenon highlighted above of increasing QMC advantage with increasing accuracy is very significant (with competing values for low #quartets and/or accuracy 70 percent but about half the MRP RF score for big #quartets and accuracy 90 percent).

3. We note further that our code is hardly optimized in that while the divide step (the maximum ratio cut) is implemented in C, the recursive algorithm is written in Perl with system calls to the MaxCut code. Thus, we are hopeful that this approach can work with even much larger data sets.

3.1.3 Comparing Experiments Characteristics

We conclude our experiments on synthetic data by demonstrating the performance of QMC (in terms of RF distance) with respect to #quartets under the two characteristics we defined: geometrical versus uniform distribution, and 70 percent versus 90 percent accuracy. The results are depicted in Fig. 12. As can be seen, the effect of the distribution (uniform versus geometrical) is more influential than the noise effect (percentage of correct quartets in the input). As shown, the two geometrical curves are significantly better (smaller RF) than the uniform reciprocals (uni 90 percent, uni 70 percent), achieving small error rates already at n^2 quartets. It appears, however, that for very large number of quartets, the uniform distribution outraces the geometrical but the significance of these numbers is not clear. Running times grow linearly with the number of quartets with 50 seconds for 100,000 quartets and 500 seconds for 1,000,000.

3.2 Experiments on Real Data

Quartet methods can be used also in the setting of supertrees. The input to the supertrees problem is a set of input trees not necessarily of size four (in fact, typically of bigger sizes). Therefore, here, in addition to the task of optimizing the tree constructed from the quartets, a separate task is to generate a "representative set" of quartets that will lead to the best construction of the tree. In particular, an edge high up near the root is a member in the central path of $O(n^4)$ quartets while on the other extreme, low edges near the leaves are members in only

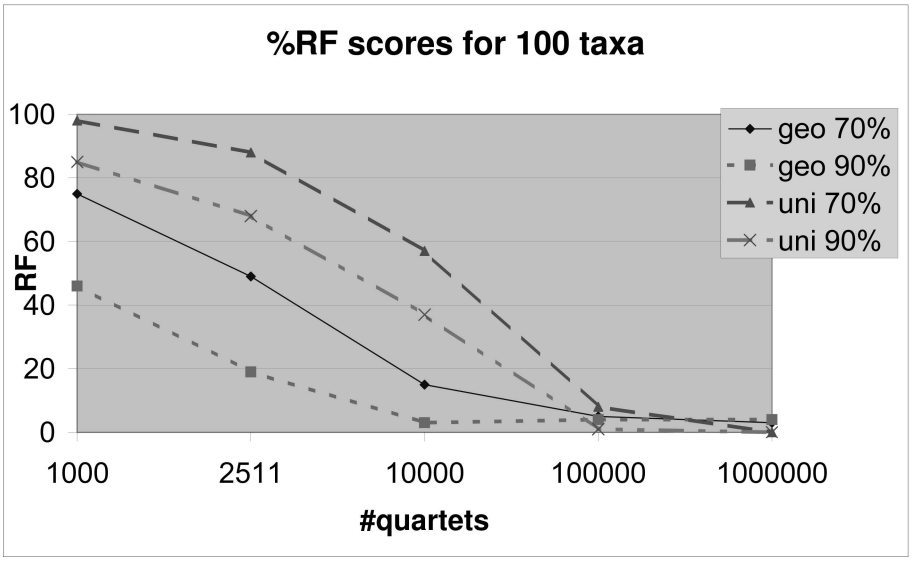

Fig. 12. A chart showing the difference in performance between the two methods (geometrical versus uniform) and the two error rates (70 percent versus 90 percent). All experiments were done for #taxa $n = 100$.

$O(n^2)$ quartets (note that pendant edges are not members in any central pass). In [29], in a rooted setting, we used a fairly naive approach of taking all $\binom{n_i}{3}$ triplets defined by subtree i with size n_i. This approach has some disadvantages: First, many of these quartets are redundant and can be eliminated. In addition, subtrees over large sets of taxa, contribute many quartets more than smaller subtrees, yielding overrepresentation of these trees. Finally, for some inputs, the amount of input quartets generated becomes so large that the running time of our method becomes very slow. Here, we devise a more advanced technique to cope with the above problems: We take a quartet q to the representative set with probability proportional to $c^{-diam(q)}$ where c is some constant calibrated according to the size of the trees/taxa set. This inverse proportion between the probability of being chosen and size of the tree spanned by a quartet, causes many quartets that define the same split to be chosen with lower probability. Since this strategy can lead to the phenomenon that some splits in a tree will remain unrepresented, we augment this set of quartets with all the short quartets of every input subtree.

In order to evaluate a supertree T with respect to an input subtree t, we obtain $T_{\mathcal{L}(t)}$, the subtree of T induced by the leaves of t. We compute the following measures:

1. the number of common edges normalized by the number of edges in $T_{\mathcal{L}(t)}$ and
2. the MAST score between the trees normalized by the number of taxa.

We comment that similar criteria are also mentioned in [12]; however, they also suggest the quartet fit measure that we found inappropriate as explained above.

3.2.1 The Data

We compared our method to MRP on five different sources of data. The first three data sets were taken from recent studies and contain data from various sources:

- **Marsupials**: A total of 158 source trees with 267 marsupial species [6]. The source trees were based on a wide range of data types, including molecular sequences, DNA hybridization, karyotypes, and immunological, morphological, and behavioral data.
- **Cetartio**: A total of 201 trees over 290 species comprised of whales, dolphins, and even-toed hoofed mammals [23].
- **Legumes**: A total of 20 trees over 571 taxa. This group contains many of the economically important and familiar temperate legumes, such as alfalfa, clovers, lentils, peas, vetches, chickpeas, licorice root, and locoweeds, as well as many of the model systems used for studies of nitrogen fixation, legume biology/genomics, and bacterial-plant symbioses/coevolution [22].

The above trees contain very big subtrees and one subtree in the Marsupial data set is over the complete taxa set. To contrast this characteristic with small input subtrees, we included two other data sets drawn from the RBCL 500 data set [27] of 500 taxa. The first is composed of 28 subtrees where each subtree is of size at most 50 taxa and the second input is of 199 trees each over at most 10 taxa.

3.2.2 Real Data Results

From each subtree in the input set, we took every quartet q with probability $1.5^{-diam(q)}$. To these we added all the short quartets of every subtree in the input. Since this set is quite sparse for such a large set, it is prone to the local components phenomenon described in Section 2.6. Therefore, we employed the quartet algorithm in conjunction with the connected components enhancement (see Section 2.6.1), in order to infer the tree. The results are shown in Table 3. The left columns represent the data set, the number of species in the data set, and the size of the largest tree. The three right double columns depict the results obtained for (from left) MRP, QMC, and QMC with the connected component feature. For each of the three first data sets, we also took subsets of the input trees where each tree in the

TABLE 3
Real Data Results

data set	#taxa	maximum tree size	MRP		QMC		QMC-conn	
			RF	MAST	RF	MAST	RF	MAST
Marsupials	267	267	62	64.1	56.5	64.6	-	-
		30	63.6	63.3	45	58.6	49	65.1
		10	74.7	63.5	68.7	65.5	71.2	69.4
Cetartio	289	212	54	65	48.2	60.6	48.2	60.6
		15	60	62.3	53	65.9	56.8	67
		10	64.1	62.1	60.7	67.6	61.7	65.5
		8	69.4	60.3	63.5	67.7	66	69
Legumes	570	140	70.2	58.8	54.3	53.3	55.3	56
		40	94.6	66.9	90.3	70	88	62
RBCL1	500	50	83.6	63.7	81.1	62.9	88.4	66.4
RBCL2	500	10	82	60	79.4	65.1	90	61.7

data set is of some bounded size. This enables us to see how the performance of each method changes when the input trees' sizes change.

In general, it seems that the connected component feature is helpful in the task of supertree construction as it yields better MAST and RF scores. As to the comparison with MRP, it seems that, in general, our method outperforms MRP on the MAST distance, especially on the smaller trees and with connected component feature. On the RBCL data, our advantage is clear and significant in both measures. A possible explanation might be that the subtrees in these two data sets are of low diameter, that is, the evolutionary distance between every pair of species in a subtree is small. Similarly to the geometrical distribution approach, this approach favors quartets of low diameter.

4 Conclusion and Further Work

In this work, we devised a quartet-based phylogenetic reconstruction method. The method is based on a recursive divide and conquer algorithm that seeks to maximize the ratio between satisfied and violated quartets at each step. This task is performed by a very fast SDP-like heuristic for solving MaxCut in a graph. We showed experimentally that although optimizing our criterion implies solving a NP-hard problem, a simple heuristic suffices to provide good performances.

We emphasize that we do not claim that our approach maximizes the number of consistent quartets or that the resulted tree is the optimal. This is beyond the scope of this paper and can be pursued in further theoretical research.

We applied this method on synthetic data in the form of quartets as well as on real data in the form of input to the supertree problem. Our results show a strict advantage over the well-known and popular supertree method MRP on the synthetic quartets input. This strict advantage was not obtained in our previous work on triplets in which we were superior over MRP in triplet score and running time, but not in the MAST and RF rates. We attribute this achievement to the parametric search algorithm that finds a near optimal ratio cut.

On the real data input, we devised a strategy of evenly picking representative quartets from the input subtrees and coping with the locality of sparse quartets over the sought phylogeny. Using these techniques, we obtained good results in comparison to MRP, when the input trees preserve locality. We hypothesize that this is related to the superiority of the geometrical distribution over uniform distribution in the tree similarity criteria.

We conclude that the use of our quartet method for supertree reconstruction seems promising mainly for its speed. However, when the input trees are very big, a deeper insight of how to represent the splits of the input subtrees is required in order to achieve clear advantage over MRP. MRP is sensitive both to the number of taxa and number of quartets whereas QMC is mainly sensitive to the number of

quartets. This gives hope that with a smart quartet selection criterion, we can reconstruct fairly large trees.

APPENDIX

Aho et al. Recursive Algorithm for Rooted Triplets

The algorithm uses two generic partitioning rules, of which only one is used in our case. It proceeds recursively on the set of taxa by applying the partitioning rule to produce a tree. The algorithm is described below:

Aho *et. al.* $(V, \mathcal{T})$

1. Let V be the set of taxa.
2. If $\mathcal{T} = \emptyset$ return a tree of depth 1 with all V as sister taxa.
3. Build the connectivity graph $G_C = (V', E')$ as follows:
   - $V' \leftarrow V$,
   - for every triplet $i, j|k \in \mathcal{T}$, introduce the edge $(i, j) \in E'$.
4. Let c be the number of connected components in G_C.
5. If $c = 1$, return NULL, no tree is consistent with all triplets.
6. else
   - create an internal vertex u in T.
   - For every connected component C_i in G,
     - $T_i \leftarrow$ Aho *et. al.* $(V(C_i), \{(i, j|k) \in \mathcal{T} : i, j, k \in V(C_i)\})$.
     - make T_i a child of u.
7. return T_u.

The Heuristic MaxCut Algorithm

Heuristic MaxCut $(G = (V, E))$

- embed V onto the 3 dimensional sphere.
- until some threshold is reached
  - find a vertex v which is *far* from the center of mass with respect to the distances induced by the graph G.
  - move v towards the center of mass.
- select a random hyperplane to partition the points.
- produce the corresponding cut.

ACKNOWLEDGMENTS

The authors would like to thank very much Tandy Warnow for insightful comments on an earlier version and also Usman Roshan for providing the RBCL data. They are also very thankful to the three anonymous referees who gave many extremely helpful comments in particular regarding the simulation study and former *TCBB* editor in chief Dan Gusfield for comments on structure and wording. The research was done at UC Berkeley and supported by NIH Grant R01-HG02362-02 and US NSF Grant CCR-0105533. Satish Rao was supported by US NSF Award-0331494.

REFERENCES

[1] A.V. Aho, Y. Sagiv, T.G. Szymanski, and J.D. Ullman, "Inferring a Tree from Lowest Common Ancestors with an Application to the Optimization of Relational Expressions," *SIAM J. Computing*, vol. 10, no. 3, pp. 405-421, 1981.

[2] S. Arora, S. Rao, and U. Vazirani, "Expander Flows, Geometric Embeddings and Graph Partitioning," *Proc. Symp. Foundations of Computer Science (FOCS)*, pp. 222-231, 2004.

[3] B.R. Baum, "Combining Trees as a Way of Combining Data Sets for Phylogenetic Inference," *Taxon*, vol. 41, pp. 3-10, 1992.

[4] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg, "Constructing Phylogenies from Quartets: Elucidation of Eutherian Superordinal Relationships," *J. Computational Biology*, vol. 5, no. 3, pp. 377-390, 1998, earlier version appeared in *Proc. RECOMB*, 1998.

[5] V. Berry and O. Gascuel, "Inferring Evolutionary Trees with Strong Combinatorial Evidence," *Theoretical Computer Science*, vol. 240, pp. 271-298, 2001.

[6] M. Cardillo, O.R.P. Bininda Emonds, E. Boakes, and A. Purvis, "A Species-Level Phylogenetic Supertree of Marsupials," *J. Zoology*, vol. 264, no. 1, pp. 11-31, 2004.

[7] M. Casanellas and J. Fernández-Sánchez, "Performance of a New Invariants Method on Homogeneous and Nonhomogeneous Quartet Trees," *Molecular Biology and Evolution*, vol. 24, no. 1, pp. 288-293, 2007.

[8] M. Casanellas, L.D. Garcia, and S. Sullivant, "Catalog of Small Trees," *Algebraic Statistics for Computational Biology*, L. Pachter and B. Sturmfels, eds., chapter 15, pp. 291-305, Cambridge Univ. Press, 2005.

[9] B. Chor, M. Hendy, B. Holland, and D. Penny, "Multiple Maxima of Likelihood in Phylogenetic Trees: An Analytic Approach," *Molecular Biology and Evolution*, vol. 17, no. 10, pp. 1529-1541, 2000, earlier version appeared in *Proc. RECOMB*, 2000.

[10] B. Chor, A. Khetan, and S. Snir, "Maximum Likelihood Molecular Clock Comb: Analytic Solutions," *J. Computational Biology*, vol. 13, pp. 819-837, 2006, Earlier Version Appeared in *Proc. Seventh Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB '03)*.

[11] B. Chor and S. Snir, "Molecular Clock Fork Phylogenies: Closed Form Analytic Maximum Likelihood Solutions," *Systematic Biology*, vol. 53, pp. 963-967, Dec. 2004.

[12] C.J. Creevey and J.O. McInerney, "Clann: Investigating Phylogenetic Information through Supertree Analyses," *Bioinformatics*, vol. 21, no. 3, pp. 390-392, 2005.

[13] C. Daskalakis, C. Hill, A. Jaffe, R. Mihaescu, E. Mossel, and S. Rao, "Maximal Accurate Forests from Distance Matrices," *Proc. 10th Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB '06)*, 2006.

[14] P. Erdös, M. Steel, L. Szekely, and T. Warnow, "A Few Logs Suffice to Build (Almost) all Trees (i)," *Random Structures and Algorithms*, vol. 14, pp. 153-184, 1999.

[15] P. Erdös, M. Steel, L. Szekely, and T. Warnow, "A Few Logs Suffice to Build (Almost) all Trees (ii)," *Theoretical Computer Science*, vol. 221, pp. 77-118, 1999.

[16] O. Eulenstein, D. Chen, J.G. Burleigh, D. Fernández-Baca, and M.J. Sanderson, "Performance of Flip Supertrees with a Heuristic Algorithm," *Systematic Biology*, vol. 53, no. 2, pp. 299-308, 2004.

[17] M.R. Garey and D.S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[18] M.X. Goemans and D.P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming," *J. Assoc. for Computing Machinery*, vol. 42, no. 6, pp. 1115-1145, Nov. 1995.

[19] I. Gronau, S. Moran, and S. Snir, "Fast and Reliable Reconstruction of Phylogenetic Trees with Very Short Branches," *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pp. 379-388, 2008.

[20] D. Huson, S. Nettles, and T. Warnow, "Disk-Covering, a Fast Converging Method for Phylogenetic Tree Reconstruction," *J. Computational Biology*, vol. 6, pp. 369-386, 1999.
[21] E. Mossel, "Distorted Metrics on Trees and Phylogenetic Forests," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 4, no. 1, pp. 108-116, Jan.-Mar. 2007.
[22] M. Wojciechowski, M. Sanderson, K. Steele, and A. Liston, "Molecular Phylogeny of the 'Temperate Herbaceous Tribes' of Papilionoid Legumes: A Supertree Approach," *Advances in Legume Systematics, Part 9*, P. Herendeen and A. Bruneau, eds., vol. 9, pp. 277-298, Royal Botanic Gardens, 2000.
[23] S. Price, O. Bininda Emonds, and J. Gittleman, "A Complete Phylogeny of the Whales, Dolphins and Even-Toed Hoofed Mammals (Cetartiodactyla)," *Biological Rev.*, vol. 80, no. 3, pp. 445-473, 2005.
[24] M.A. Ragan, "Matrix Representation in Reconstructing Phylogenetic-Relationships among the Eukaryotes," *Biosystems*, vol. 28, pp. 47-55, 1992.
[25] B. Rannala, J.P. Huelsenbeck, Z. Yang, and R. Nielsen, "Taxon Sampling and the Accuracy of Large Phylogenies," *Systematic Biology*, vol. 47, pp. 702-710, 1998.
[26] V. Ranwez and O. Gascuel, "Quartet-Based Phylogenetic Inference: Improvements and Limits," *Molecular Biology and Evolution*, vol. 18, pp. 1103-1116, 2001.
[27] K. Rice, M. Donoghue, and R. Olmstead, "Analyzing Large Datasets: *rbcL* 500 Revisited," *Systematic Biology*, vol. 46, no. 3, pp. 554-563, 1997.
[28] U. Roshan, B.M.E. Moret, T.L. Williams, and T. Warnow, "Rec-i-dcm3: A Fast Algorithmic Technique for Reconstructing Large Phylogenetic Tree," *Proc. IEEE Computational Systems Bioinformatics Conf. (CSB)*, 2004.
[29] S. Snir and S. Rao, "Using Max Cut to Enhance Rooted Trees Consistency," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 3, no. 4, pp. 323-333, Oct.-Dec. 2006, preliminary version appeared in *Proc. WABI*, 2005.
[30] S. Snir, T. Warnow, and S. Rao, "Short Quartet Puzzling: A New Quartet-Based Phylogeny Reconstruction Algorithm," *J. Computational Biology*, vol. 1, no. 15, pp. 91-103, 2008.
[31] M. Steel, "The Complexity of Reconstructing Trees from Qualitative Characters and Subtress," *J. Classification*, vol. 9, no. 1, pp. 91-116, 1992.
[32] K. Strimmer, N. Goldman, and A. von Haeseler, "Bayesian Probabilities and Quartet Puzzling," *Molecular Biology and Evolution*, vol. 14, pp. 210-211, 1997.
[33] K. Strimmer and A. von Haeseler, "Quartet Puzzling: A Quartet Maximum-Likelihood Method for Reconstructing Tree Topologies," *Molecular Biology and Evolution*, vol. 13, no. 7, pp. 964-969, 1996, ftp://ftp.ebi.ac.uk/pub/software/unix/puzzle/.
[34] S. Willson, "Building Phylogenetic Trees from Quartets by Using Local Inconsistency Measures," *Molecular Biology and Evolution*, vol. 16, pp. 685-693, 1998.

Sagi Snir received the BA degree from Bar Ilan University at Israel majoring in computer science and economics. He received the MSc and PhD degrees in computer science from the Technion, Israel. After receiving his PhD degree, he spent two years as a postdoctoral researcher in the computer science and math departments at the University of California at Berkeley. He is now at the Institute of Evolution at Haifa University, Israel. Before working on the PhD, he worked in various information technologies companies, including IBM Haifa Research Lab. His main research interest is computational biology and, in particular, phylogenetics.

Satish Rao received the BS degree from the Massachusetts Institute of Technology (MIT) in electrical engineering, computer science, and physics. He received the SM and PhD (1989) degrees in computer science from MIT. He is a professor of computer science at the University of California at Berkeley. He then held a postdoctoral position at Harvard College for a year before becoming a research scientist at the NEC Research Institute in 1991. In January 1999, he joined Akamai as a senior research scientist.