JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 15, Number 1, 2008

Pp. 91–103
DOI: 10.1089/cmb.2007.0103

Short Quartet Puzzling: A New Quartet-Based Phylogeny Reconstruction Algorithm

SAGI SNIR,[1] TANDY WARNOW,[2] and SATISH RAO[3]

ABSTRACT

Quartet-based phylogeny reconstruction methods, such as Quartet Puzzling, were introduced in the hope that they might be competitive with maximum likelihood methods, without being as computationally intensive. However, despite the numerous quartet-based methods that have been developed, their performance in simulation has been disappointing. In particular, Ranwez and Gascuel, the developers of one of the best quartet methods, conjecture that quartet-based methods have inherent limitations that make them unable to produce trees as accurate as neighbor joining or maximum parsimony. In this paper, we present *Short Quartet Puzzling*, a new quartet-based phylogeny reconstruction algorithm, and we demonstrate the improved topological accuracy of the new method over maximum parsimony and neighbor joining, disproving the conjecture of Ranwez and Gascuel. We also show a dramatic improvement over Quartet Puzzling. Thus, while our new method is not compared to any ML method (as it is not expected to be as accurate as the best of these), this study shows that quartet methods are not as limited in performance as was previously conjectured, and opens the possibility to further improvements through new algorithmic designs.

Key words: algorithms, biology, computational molecular biology, evolution.

1. INTRODUCTION

QUARTET-BASED METHODS were initially proposed in the hope that they would provide an alternative to heuristics for maximum likelihood, which are computationally intensive. These methods for phylogeny reconstruction generally have two steps: first, trees on all four-leaf subsets of the taxa are estimated (typically using maximum likelihood), and then combined in a second step into a tree on the full set of taxa. The class of quartet-based methods includes Quartet Puzzling (Strimmer and von Haeseler, 1996; Strimmer et al., 1997) and its variants (including Weight Optimization [Ranwez and Gascuel, 2001]), as well as theoretically inspired methods like Quartet Cleaning (Berry et al., 1999) and the Short Quartet Methods (Erdös et al., 1997).

While these quartet methods have theoretical guarantees (e.g., they are statistically consistent under GTR and other standard sequence evolution models) and tend to have computationally efficient algorithms,

[1]Mathematics Department, University of California, Berkeley, California.
[2]Computer Science Department, University of Texas, Austin, Texas.
[3]Computer Science Division, University of California, Berkeley, California.

their use in practice has not lived up to the original dreams. To begin with, maximum likelihood methods have improved dramatically in performance (both accuracy and speed) since the original introduction of quartet methods, with methods such as RAxML (Stamatakis et al., 2005) and GARLI (Zwickl, 2006) now able to analyze thousands of sequences in reasonable time periods (up to a few days of analysis), and producing trees that are more accurate than those recovered by NJ or maximum parsimony. However, even the relative performance of quartet methods and maximum parsimony has been discouraging. For example, in 2001, St. John et al. (2001) showed that Quartet Puzzling outperformed other quartet methods available at the time (including Quartet Cleaning) but was worse than neighbor joining. Also in 2001, Ranwez and Gascuel (2001) presented a new quartet method, called Weight Optimization (WO), based upon a novel quartet tree amalgamation technique, and demonstrated through a simulation study that WO outperformed QP. However, as Ranwez and Gascuel said in their paper, "Nevertheless, the performance of WO is still worse than those of the other tested methods. WO is always less accurate than FASTDNAML and BIONJ, and the only cases in which WO is ranked better than DNAPARS correspond to high evolution rates ... for which the parsimony methods are well known to be poorly suited." Thus, as Ranwez and Gascuel said, quartet methods may have inherent limitations in that they are "highly sensitive to 4-tree inference errors," and thus will have to be very careful in how they handle cases where some 4-trees are incorrectly inferred. Ranwez and Gascuel conjectured that "the weaknesses of the various quartet methods tested in [their] paper are very likely due to the method of weighting the 4-trees, rather than the method of combining them," and specifically pointed out the need for careful handling of quartets undergoing long-branch attraction.

Indeed, Ranwez and Gascuel studied several different weighting techniques, and none of them produced acceptable performance as compared to distance methods or maximum parsimony. However, all the weighting techniques they studied required that the weights of the three possible binary 4-trees on each quartet of taxa sum to 1.

Thus, Ranwez and Gascuel concluded that quartet methods, far from being adequate replacements for maximum likelihood methods, have inherent limitations making them less accurate than neighbor joining and maximum parsimony. Evaluating this statement is the purpose of this study.

In this paper, we present *Short Quartet Puzzling,* a novel quartet-based method which gives better estimates of the true tree topology by comparison to both neighbor joining and maximum parsimony (and very dramatically outperforms Quartet Puzzling), and hence is able to surmount the difficulties preventing the previous quartet-based methods from performing well. We estimate 4-trees using maximum likelihood, and we use a heuristic to combine the 4-trees into a tree on the full dataset, while attempting to maximize the number of 4-trees that are satisfied. However, we differ from previous techniques in two significant ways. First, we do *not* base the construction of the full tree on the entire set of 4-trees, but rather only on a subset of the possible 4-trees, which are deemed most likely to be correctly estimated (thus, we try not to include *any* 4-tree on a quartet which seems difficult to analyze correctly). We then seek to combine these 4-trees into a tree on the full dataset which satisfies a maximum number of quartets. The problem of finding a tree satisfying a maximum number of quartets is computationally difficult, and so we use an algorithm which we call *Quartet Max Cut (QMC)* which repeatedly finds maximum cuts (see below for the definition) in a graph we construct. Quartet Puzzling and Weight Optimization also try to optimize the same criterion, but use different heuristics than we use in QMC. We provide a brief description of QMC in this paper, but the full description is provided in Snir and Rao (2007), which also gives some of its mathematical properties.

The specific technique we use for the determination of the quartets we will include in our amalgamation phase is based upon the observations in Erdos et al. (1999a, 1999b) and Huson et al. (1999), which showed that quartets with smaller evolutionary diameters are more likely to be accurately estimated than quartets with larger diameters. This observation led us to develop a selection technique which is biased in favor of quartets with smaller diameters rather than larger diameters. This selection technique also uses randomness to reduce the number of quartets that are selected, thereby keeping the running time reasonable. The combined approach reduces the probability of including quartets which are likely to be poorly analyzed, and also reduces the incidence of "long branch attraction."

The resultant two-phase technique thus has two novel algorithmic components (the randomized technique for selecting quartets, and the Quartet Max Cut [QMC] technique for amalgamating quartet trees together); however, most importantly, it breaks the barrier previously limiting quartet methods.

Thus, although our method is unlikely to improve upon the current best ML methods, such as RAxML or GARLI, this study does provide evidence that it *is* possible to design quartet-based methods that can outperform both MP and NJ, which was conjectured by Ranwez and Gascuel to be an upper limit on the performance of all quartet methods. Furthermore, the new method is *dramatically* better than Quartet Puzzling, showing the significance of this improvement.

In the rest of the paper, we describe our method in detail, and present the results of a simulation study comparing our new method to Quartet Puzzling, neighbor joining, and maximum parsimony.

2. SHORT QUARTET PUZZLING

We begin by defining quartet trees: A *quartet tree* is a tree on four leaves $\{a, b, c, d\}$ (Fig. 1). We write a quartet tree t over $\{a, b, c, d\}$ as $ab|cd$ if there exists an edge in t splitting a, b from c, d. We will assume that all quartet trees are fully resolved (i.e., binary) trees, and so have an edge separating two of the leaves from the other two. Hence, each binary quartet tree on a, b, c, d can be written as one of $ab|cd, ac|bd$, or $ad|bc$. When a tree T on the set S of taxa has an edge separating a, b from c, d, then we say that T induces the quartet tree $ab|cd$, and that the quartet tree $ab|cd$ is *consistent* with T.

Our quartet method has three steps:

Step 1: Given an input set S of n sequences in an alignment, the method first selects a set Q of four-taxon subsets for which it will estimate trees.

Step 2: For each element $q \in Q$, we use maximum likelihood to estimate the tree, thus producing a quartet tree t_q on q.

Step 3: We apply our heuristic amalgamation method to combine the collection of quartet trees $\mathcal{T}(Q) = \{t_q : q \in Q\}$ into a tree T on the set S.

We now elaborate on these steps.

Step 1: Selecting the subset of four-taxon subsets. Given a set S of aligned sequences, we first compute a "guide tree" T_g on the full set of sequences. T_g need not be particularly accurate, since its sole purpose is to determine those four-taxon subsets for which we will estimate maximum likelihood trees. (In our experiments, we computed a neighbor joining tree—using PAUP*'s implementation—on the set S, using the Jukes-Cantor distance correction [Jukes and Cantor, 1969], but other methods can also be used.)

Once we have the guide tree, T_g, we use an additional technique to define the set Q of quartets on which to estimate trees. The technique we use is the following:

- We treat T_g as a weighted tree, by letting every edge have length 1. We then compute, for every quartet q of taxa, the diameter of q, which we denote by $diam(q)$. (When all edges have unit weight, then the diameter of q is also called the "topological diameter" since it equals the number of edges in the longest path between any two taxa in q.) Other edge-weighting techniques could also be used, including the edge lengths computed by NJ.
- We include q in Q with probability $b^{-diam(q)}$, for a properly chosen $b \geq 1$.

Selecting the optimal base b. Using the function $b^{-diam(q)}$ to denote the probability of choosing a quartet q provides us with a lot of flexibility. Note that the set Q depends upon b, and that different values of b give rise to different subsets of quartets: for the special case $b = 1$ we obtain the full set of quartets,

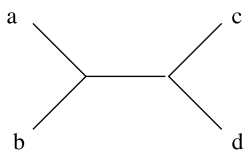

FIG. 1. The quartet tree $((a, b), (c, d))$.

but for larger values of b the set Q is likely to miss some quartets. Furthermore, the larger that b is, the smaller the set Q is likely to be. For all $b > 1$, the set Q is more likely to include a quartet if its diameter in the guide tree is small, and less likely to include a quartet whose diameter in the guide tree is large. Therefore, if we let b be small, then Q will contain many quartets, increasing the probability of including incorrect quartet trees (Erdos et al., 1999a). Having b be large, however, produces a different problem: although the quartet trees are more likely to be correct, there may not be enough coverage to uniquely determine the tree. The selection of b thus can have a large impact on the algorithm. We experimented with different values of b and selected $b = 1.2$ since the topological accuracy of the resultant trees was better at this value than at the neighboring values we examined. Table 1 (in the Appendix) shows the performance of our method under three values for b: 1, 1.2 and 1.5, for datasets containing between 30 and 50 taxa. Note that $b = 1.2$ generally gives better results than setting $b = 1$ or $b = 1.5$. Note that $b = 1$ generates the additional technical drawback of producing large inputs, which impacts the running time of the method. Based on these results we selected $b = 1.2$ for all our experiments.

Step 2: Computing the trees on each quartet. At this point, we have a set Q of quartets, and we construct the ML tree for each quartet in Q. Thus, we obtain the set $\mathcal{T}(Q)$, consisting of the maximum likelihood trees on each quartet in Q.

Step 3: Computing the supertree from the quartet trees. The input to this step of the method is a set $\mathcal{T}(Q)$ of quartet trees, which may or may not be compatible with any supertree, and the objective is the construction of a tree which maximizes the number of satisfied quartet trees. This is a NP hard problem, and so rather than attempting to solve this problem provably correctly (which would likely require exponential time), we will instead seek to develop a heuristic which performs well in practice and which does not take too long.

Quartet Puzzling and Weight Optimization use heuristics for this optimization problem which build a tree one leaf at a time, on the basis of having trees for all quartets of taxa. The heuristic we developed was designed so that it would not need to have trees on all quartets. The technique we use is similar in spirit to the technique developed by Semple and Steel (2000) for the corresponding problem when the input consists of rooted subtrees; there, however, the computational problem of determining if a supertree exists that is compatible with all the input subtrees is easier than ours. The commonality of the two methods lies in how both methods handle evident conflicts in the input subtrees: both use a divide-and-conquer technique, whereby the set of taxa is divided into two sets (so as to produce a split that would then appear in the tree that we construct) while minimizing the number of violated subtrees. Our technique is more complicated than Semple and Steel's technique, reflecting the relative difficulty in reconstructing trees from unrooted subtrees as compared to rooted subtrees. Our division of the dataset into subsets is obtained through the calculation of a maximum cut (MaxCut) (Garey and Johnson, 1979) (a subset of the edges whose removal separates the graph into two parts, such that the total weight of the edge set is maximum) in a graph representing the input set of quartets.

Quartet MaxCut (QMC). In this section, we provide a high-level description of the algorithm, quartet MaxCut, or "QMC." Full details, including mathematical properties related to it, can be found in a companion paper (Snir and Rao, 2007). We first give an intuitive high-level description of how we graphically represent our input, so that our MaxCut algorithm will, in a given iteration, continue to satisfy a large (hopefully maximum) number of the input quartets.

QMC constructs a graph based upon the set of input quartets, so that the taxa are represented by nodes in the graph and the quartets are represented by edges (called "good" and "bad" edges). A partition of the vertices of the graph into two parts can thus be seen as defining a recursion for the QMC algorithm on the dataset, with the implication being that the constructed tree will contain an edge inducing the same bipartition on the leaf set. Thus, QMC partitions a set of nodes into two parts, recursively solves the subproblems induced by each part (with the appropriate set of quartet trees), and merges the solutions obtained into a tree on the full set.

In the tree that is constructed, there will be an edge e that separates the leaves in one part of the bipartition from the other part. Clearly, a quartet that is contained entirely on one side of a bipartition is

not affected by it, but other quartets *are* affected. Those that split 2/2 across the bipartition will either be violated or satisfied by the bipartition; those that split 3/1 will be "deferred." We say:

- An affected quartet $q = ((a, b), (c, d))$ that is split 2/2 is *satisfied* if a and b are both on the same side of the bipartition. Otherwise, q is *violated by* the bipartition (i.e., a and c or a and d are in the same side of the bipartition).
- For all other affected quartets q, we have only a single taxon in some part of the bipartition 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. At the end of the process a tree is reconstructed, and every quartet is either satisfied or violated. Therefore, a plausible strategy for the decomposition technique with respect to the overall goal of minimizing the number of violated quartets is to maximize the *ratio* between satisfied and violated quartets at every step.

We now describe the graph construction. Given the set of quartets Q over the taxon set S, we build the weighted graph $G = G(Q) = (V, E)$ with $V = S$ and edge set E, as follows. We associate every taxon in S with a vertex in the graph G, and for every quartet $q = ab|cd \in Q$ we add the edges $\{(a, c), (a, d), (b, c), (b, d), (a, b), (c, d)\}$ to the graph. Edges are either "good edges" or "bad edges," with edges $(a, c), (a, d), (b, c)$, and (b, d) the *good edges* and (a, b) and (c, d) the *bad edges* (Fig. 2). Observe that an edge that is good for one quartet can be bad for another, and that the graph $G(Q)$ can have parallel edges.

A *cut* in the graph is a subset E_0 of the edges so that the removal of the edges in E_0 (but not the endpoints) separates the vertices of the graph into two parts. Since the vertices of our graph represent taxa, a cut in the graph thus corresponds to a partition of the set of taxa. Given a cut in the graph constructed above, an affected quartet contributes four good edges and no bad edges to the cut if the quartet is satisfied, two good edges and one bad edge if the edge is deferred, or two good edges and two bad edges if it is violated.

This establishes the relationship between the number of quartets satisfied, violated, and deferred, and the number of good and bad edges in the cut and motivates the use of the recursive MaxCut at every recursive step.

The second part of the high level algorithm occurs upon the return from the recursive step. In contrast to the rooted setting where every triplet is either satisfied, violated or wholly contained in one of the parts here a quartet can be deferred to the successive step without being contained in any of the parts. The challenge is to keep track of these quartets and try to satisfy them.

We use the following approach to cope with the above hurdles: At every step of the algorithm a max cut is computed, and each part is solved recursively but with some modifications. Suppose that the division of the dataset is into two sets A and B. We create two artificial taxa, x_A and x_B, for A and B, respectively. Now suppose that the quartet $uu'|vv'$ is deferred, because $u \in A$ and $\{u', v, v'\} \subset B$. We replace $uu'|vv'$ by quartet by $x_A, u'|v, v'$, and we add this quartet to the set of quartets that are applicable to B; thus, in essence, we treat B as including x_A. After returning from the recursions on both parts, the tree we obtain for A includes the taxon x_B, and the tree we obtain for B includes the taxon x_A. These trees are then joined into a tree on the leaf set $A \cup B$. trees for each part are joined by attaching x_A and x_B with an edge, and then suppressing these degree two nodes (for an illustration of this process, see Fig. 3).

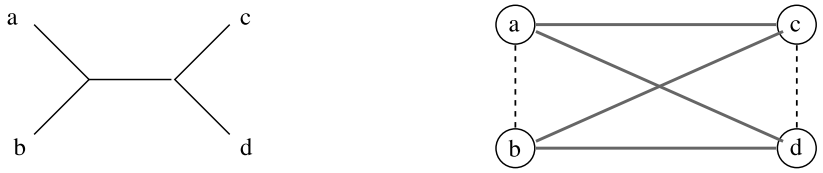

FIG. 2. For a quartet $ab|cd$ (left), we draw the four good edges (solid) $\{(a, c), (a, d), (b, c), (b, d)\}$ and the two bad edges (dashed) $\{(a, b), (c, d)\}$ (right).

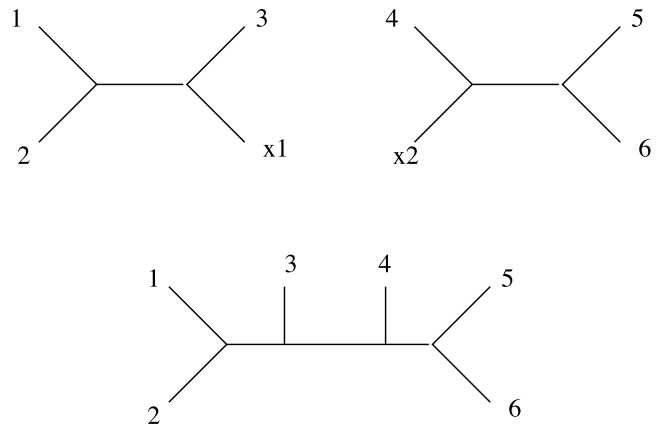

FIG. 3. The two subtrees (above) are joined by removing the two artificial nodes and connecting their corresponding two pendant edges (below).

3. METHODS

We compared our method to Quartet Puzzling (QP), neighbor joining (NJ), and maximum parsimony (MP).

Neighbor joining. We used NJ (Saitou and Nei, 1987) as implemented in PAUP*. However, we tested several different distance corrections, in order to determine the distance correction that gave the most accurate phylogenies. Our simulation study evolves sequences under the GTR, and so we included logdet (Steel, 1994) distances, but also Jukes-Cantor (JC) (Jukes and Cantor, 1969) and K2P (Kimura, 1980) distances. We chose JC distances since the topological accuracy of the trees under JC distances proved to be slightly better (under all model conditions!) than those obtained using either logdet and K2P distances (Table 3) (in the Appendix).

Maximum parsimony. We used heuristic searches in PAUP* for maximum parsimony saving the 10,000 best trees in memory. We also explored the performance of PAUP*'s heuristic search for maximum parsimony by varying the number of random sequence additions (for the values 1, 30, and 5000). In these studies, allowing a longer search had an extremely small impact on the topological accuracy of the majority consensus tree (see Table 4, given in the Appendix), and did not change the relative performance of MP compared to the other methods we explored. Thus, we used the following PAUP* command for our heuristic search:

set Warnreset = no Autoclose = yes MaxTrees = 10,000;

hsearch NReps = 30 Addseq = random;

contree all/ strict = no majrule = yes;

Quartet puzzling. Quartet puzzling (Strimmer and von Haeseler, 1996; Strimmer et al., 1997) is probably the best known quartet-based phylogenetic reconstruction method. Roughly speaking, QP has the following algorithmic structure:

- Using maximum likelihood, estimate the tree on every set of four taxa.
- For some large number of random taxon addition orders, do:
  - Begin with a star tree on the first three taxa.
  - Repeatedly insert the next taxon into the current tree so as to minimize a function of the number of violated quartets.
- Return the majority consensus tree of the trees obtained, one for each random taxon addition order.

We used the version of QP in TREE-PUZZLE 5.2 (Schmidt et al., 2002), the current parallel version of QP. We used the default setting for Quartet Puzzling, and so performed 1000 random taxon addition

orders. Since the sequences were evolved under the GTR model, we estimated ML trees on every set of four taxa under the GTR model.

Short quartet puzzling. Recall that SQP computes the ML tree on all quartets; in our experiment, we handled this by using the first phase of Quartet Puzzling in which ML trees on all quartets are computed. We then determine those quartet trees we will be included in the amalgamation step through the following process. First, we compute a guide tree using neighbor joining based upon Jukes-Cantor distances. We weight the edges of the guide tree by having all edges have unit weight, and then compute the diameter within the guide tree of each quartet of taxa; this allows us to compute the set Q of quartets that we will use in the amalgamation step. We look up the ML tree for each of our selected quartets. Finally, we applied QMC to the set of quartets we obtained, and obtained a tree on the full dataset. Note that the quartet amalgamation technique, QMC, typically, but not always, produces binary trees (though this is dependent on the number and quality of the quartet trees provided). However, MP and Quartet Puzzling typically produce unresolved trees due to the use of the majority consensus method.

4. SIMULATION STUDY

We now describe the simulation study we performed to evaluate the performance of this new method.

Simulated datasets. We used the r8s (Sanderson) software to produce random birth-death trees with varying numbers of taxa, ranging from 30 up to 80. The random birth-death trees produced by r8s are ultrametric binary trees with edge lengths so that the length of the path from the root to every leaf is the same, and which have root-to-leaf distance of 1. These edge lengths are then modified as follows. First, we deviate the tree from this strong molecular clock assumption by multiplying every edge by a number chosen randomly and uniformly in the interval $[0.85, 1.15]$. Secondly, we rescale the tree by multiplying every edge length by 0.2, to produce a model tree with a moderate maximum leaf-to-leaf distance (this avoids the portion of the parameter space for which maximum parsimony is known to perform badly). We set the remaining parameters for the GTR model of evolution using the values from Zwickl and Hillis (2002).

We generated 100 model trees for each number of taxa, and on each model tree we generated two datasets, one of length 500 and one of length 1000. This results in 1200 datasets, each of which is analyzed by the methods we studied.

Error rates. The standard practice in the field is to report Robinson-Foulds (RF) distances; however, the use of RF distances has been rightly criticized (Rannala et al., 1998), and so we elected to report False Negative (FN) rates (also called Missing Edge rates) and False Positive (FP) rates, as well as the RF error rate which is the average of these two error rates. We also computed the MAST (Maximum Agreement Subtree) 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 (Eulenstein et al., 2004).

- *False Positive (FP) rate*: The splits (edges) in the estimated tree T which do not appear in T_0 are "false positives." This value is normalized by the number of internal edges in the estimated tree, to produce a value of 0–100%.
- *False Negative (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 of 0–100%.
- *Maximum Agreement Subtree (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, 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%. Note that if the MAST distance between two trees is 0, then the two trees are identical.

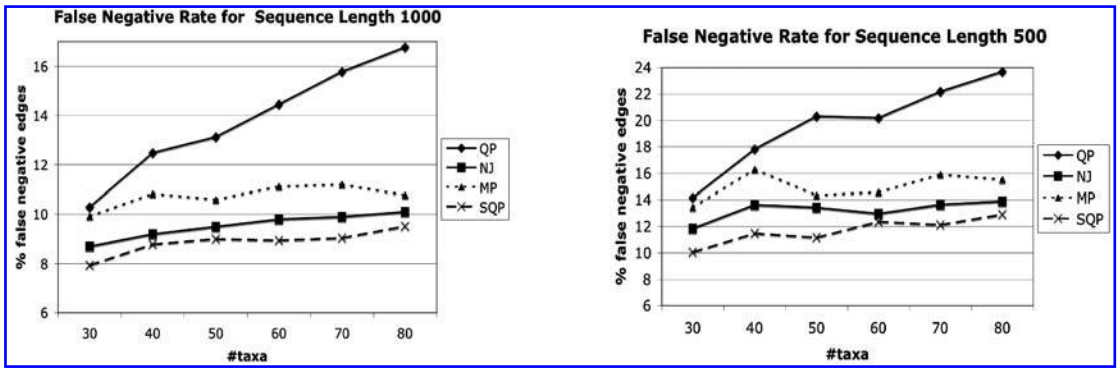

FIG. 4. False negative rates (given as percentages) for quartet puzzling (QP), neighbor joining (NJ), maximum parsimony (MP) and SQP, as a function of the number of taxa and sequence length. We simulated sequences down GTR model trees with a slight deviation from the molecular clock.

5. RESULTS

False negatives (missing edge) rates. Note that Quartet Puzzling (QP) has a *much* higher FN rate than the other methods, and the FN rate increases with the number of taxa (Fig. 4). By contrast, SQP gives the best FN rate at every sequence length and number of taxa we studied, followed by NJ and then by MP.

False positive rates. In our study, QP and MP return majority consensus trees, but SQP and NJ return binary trees. Hence, we would predict that generally QP and MP would get better FP rates than SQP or NJ. However, the results (Fig. 5), are a little surprising: with respect to false positive rates, Quartet Puzzling (QP) outperforms the other methods, and SQP is generally the next best, followed then by NJ and MP.

Robinson-Foulds error rates. Finally, we show the RF error rate (Fig. 6), which is the average of the FN and FP rates. We see that SQP is the best of the methods we compared for all dataset sizes and both sequence lengths; furthermore, the improvement over QP is dramatic, and increases with the number of taxa.

MAST distances. These results, given in Figure 7, show that SQP is by far the most accurate, QP is the worst, and that MP and NJ have about the same performance. Note that for this metric, reconstructed trees that are unresolved are automatically penalized, since polytomies are considered "hard."

Discussion of error rates. We have shown that SQP clearly improves over the other methods with respect to Robinson-Foulds, False Negative, and MAST error rates. Furthermore, QP's FN and MAST

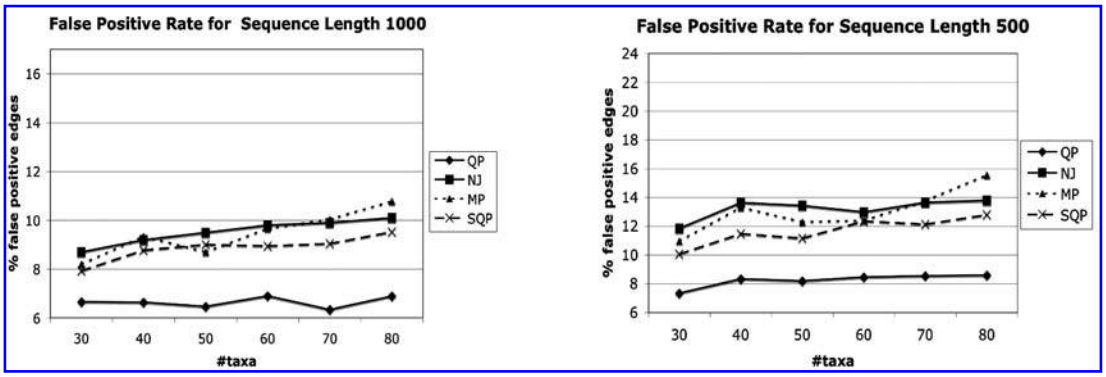

FIG. 5. False positive rates (given as percentages) for quartet puzzling (QP), neighbor joining (NJ), maximum parsimony (MP) and SQP, as a function of the number of taxa and sequence length. We simulated sequences down GTR model trees with a slight deviation from the molecular clock.

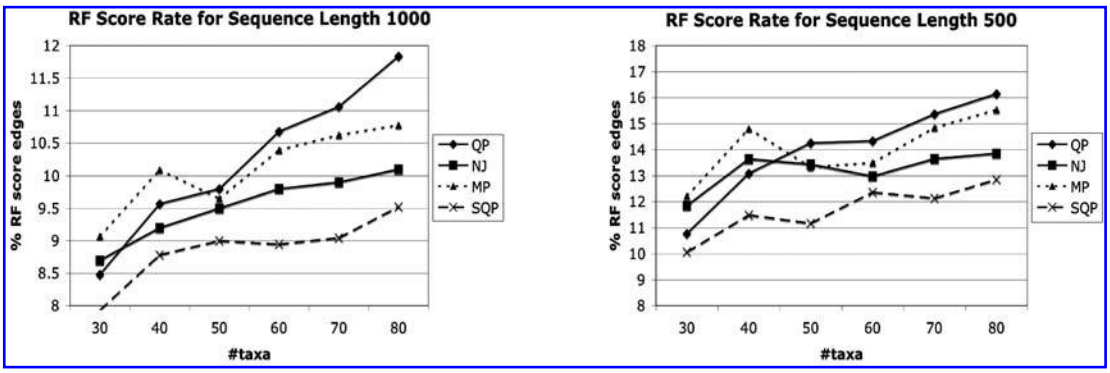

FIG. 6. The Robinson-Foulds error rate (given as percentages) for quartet puzzling (QP), neighbor joining (NJ), maximum parsimony (MP) and SQP, as a function of the number of taxa and sequence length. We simulated sequences down GTR model trees with a slight deviation from the molecular clock.

error rates increase quickly with increasing number of taxa. However, QP is the best with respect to False Positive rates, followed by SQP.

Running times. Table 2 presents the running times for the four methods we studied. Unsurprisingly, neighbor joining is the fastest of the four methods we analyzed, using at most one second on all the inputs we analyzed, maximum parsimony is slower than neighbor joining, but still fast for these datasets. The comparison we focus on here is between QP and SQP.

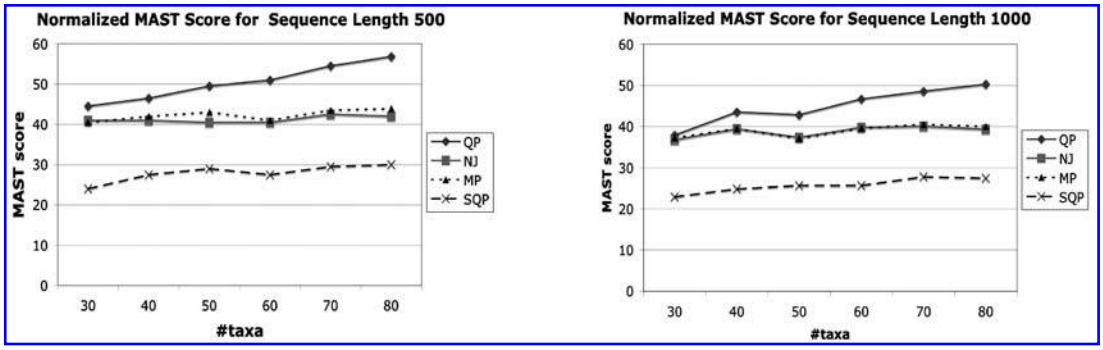

FIG. 7. MAST distances (given as percentages) for quartet puzzling (QP), neighbor joining (NJ), maximum parsimony (MP) and SQP, as a function of the number of taxa and sequence length. We simulated sequences down GTR model trees with a slight deviation from the molecular clock. NJ trees were calculated using Jukes-Cantor distances which produced the most accurate phylogenetic estimates.

TABLE 2. RUNNING TIMES FOR ALL METHODS (IN SECONDS)

					QP				*SQP*			
	NJ		*MP*		*ML*		*Puzzling*		*ML*		*QMC*	
n	*500*	*1000*	*500*	*1000*	*500*	*1000*	*500*	*1000*	*500*	*1000*	*500*	*1000*
30	1	1	6	12	17	32	36	32	17	32	7	8
40	1	1	14	20	53	101	114	101	53	101	20	19
50	1	1	32	42	93	205	314	275	93	205	51	50
60	1	1	54	63	236	619	1486	1361	236	619	91	80
70	1	1	83	99	437	1030	2796	2500	437	1030	152	149

Recall that Quartet Puzzling has two steps: first it uses ML to estimate trees on each of the possible $\binom{n}{4}$ quartets (see the two left columns in Table 2) and then it performs 1000 puzzling steps, where it combines these quartet trees into trees on the full dataset, and computes a consensus tree of these 1000 trees (see the two right columns in Table 2); both steps are clearly computationally intensive—however, note that the "puzzling" step is by far the more expensive. Thus, for 70 taxa, it takes more than 17 minutes to calculate the ML trees on all the four taxon datasets for sequences of length 1000. However, it takes almost 42 minutes for the puzzling step! Note also that the second step could be reduced by doing fewer sequence addition orders, but this is not recommended by the designers of QP.

We now discuss the running time for SQP. Recall that SQP has several steps: first we compute the guide tree, and the diameter within the guide tree for every quartet in our dataset. We then compute the set Q of quartets and look up the Quartet Puzzling tree for each of these quartets. Finally, we apply Quartet MaxCut (QMC) to these quartet trees to produce the output tree.

Therefore, we will focus on the comparison between the puzzling step for QP and the remainder of the SQP running time, which is just the call to Quartet MaxCut. These running times for SQP are given in the rightmost columns, and should be compared to the "Puzzling" step for QP.

Note the following: the quartet amalgamation step of the SQP method is always faster (in these datasets) than the quartet amalgamation step for QP, and this difference is substantial. For example, in the quartet amalgamation step for 70 sequences of length 500, QP takes approximately 45 minutes to complete its analysis, while SQP takes less than 3 minutes.

The overall running time of SQP is still high, however, because in its current implementation we are using QP to do the maximum likelihood tree estimation step. In the conclusions section, we discuss how we can further reduce the running time of SQP by eliminating this dependency on QP.

6. CONCLUSION

In this work, we devised a quartet-based phylogenetic reconstruction method, and demonstrated improved performance of this method over the best known quartet method, Quartet Puzzling. More surprising, we showed that with respect to topological accuracy, our method was able to outperform both neighbor joining and maximum parsimony which previous quartet-based methods (including Weight Optimization by Ranwez and Gascuel) had not been able to do. This improvement shows that quartet methods have greater potential than had been previously realized.

We conjecture that the performance advantage we obtain is a result of several aspects of the design of this method, including the selective use of quartets that are likely to be "short quartets" in the true tree. Our data shows that trying to construct a tree from the full set of ML quartet trees generally produces less accurate estimates of evolution, showing that some of our advantage clearly arises from how we select the quartet trees.

We turn now to a comment made by Ranwez and Gascuel (2001): "The weaknesses of the various quartet methods tested in this paper are very likely due to the method of weighting the 4-trees, rather than the method of combining them. Indeed, as explained above, considering only four taxa requires correct management of long-branch attraction." Their study examined several different ways of weighting trees on quartets, and yet none were able to break the performance barrier. However, all the weightings they considered had the property that the weights of the three trees on any given quartet sum to 1, a restriction that is not necessarily helpful. Consider, then, our technique for selecting quartet trees for inclusion in our amalgamation step. It is easy to see this technique as a weighting scheme on quartet trees, where we allow a tree to either have weight 0 or weight 1. Furthermore, on a given quartet a, b, c, d, either all the trees on a, b, c, d have weight 0 (in which case none of them is included for the quartet a, b, c, d), or for exactly one of the three trees has weight 1 and the rest have weight 0 (in which case we have included exactly one tree for the quartet a, b, c, d). Note that this way of assigning weights to trees violates the assumption that all weightings on the three trees on any given quartet of taxa sum to 1. We conjecture that one of the reasons we are able to get improved performance is because we use a more flexible weighting scheme in which the weights reflect the confidence we have in the ML tree. Thus, we conjecture that new weighting schemes which allow for weights that need not add up to 1 on every quartet and which are statistically derived would allow even greater improvements. Indeed, our weighting scheme is extremely simple, and

we strongly suspect that more sophisticated techniques would lead to improved performance, especially on the largest trees.

We now discuss the computational demands of quartet methods for phylogeny reconstruction. Any quartet method that requires the estimation of a tree on all quartets will be computationally expensive, since the number of quartets is on the order of n^4. Hence, for quartet methods to become computationally acceptable methods for phylogenetic inference on large datasets, quartet methods must be modified so that they do not depend upon estimating trees on all quartets.

The quartet method we introduced, Short Quartet Puzzling, still requires running time proportional to n^4, since it explicitly calculates the diameter of every quartet tree in the guide tree; furthermore, in its current implementation, we use Quartet Puzzling for ML tree estimation. Both these aspects of the algorithm would need to be modified for SQP to become computationally fast enough for use on large datasets. Obvious candidates for improvement include using a deterministic technique for defining Q instead of this randomized technique (which requires that we compute the diameter of all quartet trees), and we could replace the use of Quartet Puzzling to compute ML trees by a technique which only estimated ML trees for those quartets that are placed in Q. Since Q will in general contain only a sparse selection of the quartets, these two modifications would help to reduce the running time significantly. These modifications are part of our planned research.

An alpha version of the current software is available upon request from the first author.

7. APPENDIX

In this Appendix, we provide some additional data showing the results of experiments performed in order to set the variable parts of our experiment.

TABLE 1. RF AND MAST ERROR RATES FOR THREE VALUES FOR b

	$b = 1$		$b = 1.2$		$b = 1.5$	
	RF	*MAST*	*RF*	*MAST*	*RF*	*MAST*
30	8.55	23.4	7.88	22.9	11.14	25.5
40	9.29	27.8	8.75	24.8	12.24	28.3
50	8.95	26.7	8.99	25.6	12.09	27.8

Since SQP generally produces binary trees, RF error rates are equal to false negative and false positive rates.

TABLE 3. ROBINSON-FOULDS (RF) ERROR RATES FOR NJ UNDER LOGDET, JC, AND K2P DISTANCES, FOR VARYING NUMBERS OF SEQUENCES GENERATED UNDER THE GTR MODEL, EACH OF LENGTH 1000

No. of taxa	*JC*	*K2P*	*Logdet*
30	8.67	9.46	9.17
40	9.27	9.64	9.56
50	9.49	10.53	10.12
60	9.125	9.94	9.875
70	9.76	10.24	10.31
80	10.105	10.95	10.72

Since NJ returns a single tree, RF distances are equal to the False Negative and False Positive rates.

TABLE 4. FALSE NEGATIVE (FN) AND FALSE POSITIVE (FP) ERROR RATES FOR SLOW, MID, AND FAST HEURISTIC SEARCHES FOR MAXIMUM PARSIMONY, ON DATASETS WITH VARYING NUMBERS OF SEQUENCES EACH WITH 1000 SITES

False Negative (FN)				*False Positive (FP)*			
No. of taxa	*FN-slow*	*FN-mid*	*FN-fast*	*No. of taxa*	*FP-slow*	*FP-mid*	*FP-fast*
30	8.96	8.96	8.96	30	7.31	7.31	7.59
40	10.59	10.59	10.54	40	9.29	9.29	9.4
50	11.57	10.85	11.36	50	10.25	8.78	10.44
60	11.44	11.44	11.37	60	9.86	9.86	10.01
70	12.03	12.06	11.79	70	11.06	11.01	10.85
80	11.45	10.57	11.71	80	10.39	10.57	10.84

Different heuristic searches for MP. We begin with the comparison between three ways of running PAUP*'s heuristic search for maximum parsimony. These ways differ only in the number of random sequence addition orders. For the fast MP search, we had only one sequence addition order. For the "mid" MP search we had 30 sequence addition orders, and for the "slow" MP search we had 5000. In each case, we computed a majority consensus tree (Table 4).

Note that there are very small differences in topological accuracy between the slowest and the fastest versions of MP; this may reflect the fact that we always return a consensus tree. Given the very small difference in performance between the error rates of these variants, we chose the mid version for all our experiments.

Neighbor joining distance corrections. Recall that our simulation evolves sequences under GTR, and so logdet distances are a statistically consistent way of computing evolutionary distances. However, we also examined the performance of NJ using other distance corrections, to see which technique produced the topologically most accurate methods. Interestingly, although Jukes-Cantor (JC) is not a statistically consistent distance estimator for GTR evolution, NJ under JC distances is more accurate than NJ under either K2P or logdet distances. Therefore, in our experiments we reported NJ under JC distances.

ACKNOWLEDGMENTS

This research was supported by NIH grant R01-HG02362-02 and NSF grant CCR-0105533 (to S.S.), NSF award 0331453 (to T.W.), and NSF award 0331494 (to S.R.).

REFERENCES

Berry, V., Jiang, T., Kearney, P., et al. 1999. Quartet cleaning: improved algorithms and simulations. *Eur. Symp. Algorithms*, 313–324.

Erdös, P.L., Steel, M., Székely, L.A., et al. 1997. Constructing big trees from short sequences. *Lect. Notes Comput. Sci.* 1256, 827–837,

Erdos, P.L., Steel, M.A., Szekely, L., et al. 1999a. A few logs suffice to build (almost) all trees (i). *Random Struct. Algorithms* 14, 153–184.

Erdos, P.L., Steel, M.A., Szekely, L., et al. 1999b. A few logs suffice to build (almost) all trees (ii). *Theoret. Comput. Sci.* 221, 77–118.

Eulenstein, O., Chen, D., Burleigh, J.G., et al. 2004. Performance of flip supertrees with a heuristic algorithm. *Syst. Biol.* 53, 299–308.

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

Huson, D.H., Nettles, S., and Warnow, T. 1999. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *J. Comput. Biol.* 6, 369–386.

Jukes, T.H., and Cantor, C.R. 1969. Evolution of protein molecules, 21–132. In: Munro, H.N., ed., *Mammalian Protein Metabolism*. Academic Press, New York.

Kimura, M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* 17, 111–120.

Rannala, B., Huelsenbeck, J.P., Yang, Z., et al. 1998. Taxon sampling and the accuracy of large phylogenies. *Syst. Biol.* 47, 702–710.

Ranwez, V., and Gascuel, O. 2001. Quartet-based phylogenetic inference: improvements and limits. *Mol. Biol. Evol.* 18, 1103–1116.

Saitou, N., and Nei, M. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* 4, 406–425.

Sanderson, M. 2007. r8s. Available at: *http://ginger.ucdavis.edu/r8s/*. Accessed December 1, 2007.

Schmidt, H.A., Strimmer, K., Vingron, M., et al. 2002. Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics* 18, 502–504.

Semple, C., and Steel, M. 2000. A supertree method for rooted trees. *Discrete Appl. Math.* 103, 147–158.

Snir, S., and Rao, S. 2007. Quartets maxcut: a divide and conquer quartets algorithm (submitted).

St. John, K., Warnow, T., Moret, B.M.E., et al. 2001. Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. *Proc. Sixth Annu. ACM-SIAM Symp. Discrete Algorithms*, 196–205.

Stamatakis, A., Ludwig, T., and Meier, H. 2005. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics* 21, 456–463.

Steel, M.A. 1994. Recovering a tree from the leaf colourations it generates under a Markov model. *Appl. Math. Lett.* 7, 19–24.

Strimmer, K., Goldman, N., and von Haeseler, A. 1997. Bayesian probabilities and quartet puzzling. *Mol. Biol. Evol.* 14, 210–211.

Strimmer, K., and von Haeseler, A. 1996. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.* 13,964–969.

Zwickl, D. 2006. GARLI download page. Available at: *www.zo.utexas.edu/faculty/antisense/Garli.html*. Accessed December 1, 2007.

Zwickl, D., and Hillis, D. 2002. Increased taxon sampling greatly reduces phylogenetic error. *Syst. Biol.* 51, 588–598.

Address reprint requests to:
Dr. Sagi Snir
Mathematics Department
University of California
Berkeley, CA 94720

E-mail: ssagi@math.berkeley.edu

This article has been cited by:

1. Michelle R. Lacey , Jason Calmes . 2009. A Sharp Error Probability Estimate for the Reconstruction of Phylogenetic Quartets by the Four-Point MethodA Sharp Error Probability Estimate for the Reconstruction of Phylogenetic Quartets by the Four-Point Method. *Journal of Computational Biology* **16**:3, 443-456. [Abstract] [PDF] [PDF Plus]