Using Semi-definite Programming to Enhance Supertree Resolvability

Shlomo Moran[1], Satish Rao[2], and Sagi Snir[3]

[1] Computer Science dept., Technion, Haifa 32000, Israel[†]
moran@cs.technion.ac.il
[2] Computer Science dept. University of California, Berkeley, CA 94720, USA[‡]
satishr@cs.berkeley.edu
[3] Mathematics dept. University of California, Berkeley, CA 94720, USA[§]
ssagi@math.berkeley.edu

Abstract. Supertree methods are used to construct a large tree over a large set of taxa, from a set of small trees over overlapping subsets of the complete taxa set. Since accurate reconstruction methods are currently limited to a maximum of few dozens of taxa, the use of a supertree method in order to construct the tree of life is inevitable.

Supertree methods are broadly divided according to the input trees: When the input trees are unrooted, the basic reconstruction unit is a quartet tree. In this case, the basic decision problem of whether there exists a tree that agrees with all quartets is NP-complete. On the other hand, when the input trees are rooted, the basic reconstruction unit is a rooted triplet, and the above decision problem has a polynomial time algorithm. However, when there is no tree which agrees with all triplets, it would be desirable to find the tree that agrees with the maximum number of triplets. However, this optimization problem was shown to be NP-hard. Current heuristic approaches perform mincut on a graph representing the triplets inconsistency and return a tree that is guaranteed to satisfy some required properties.

In this work we present a different heuristic approach that guarantees the properties provided by the current methods and give experimental evidence that it significantly outperforms currently used methods. This method is based on divide and conquer where we use a semi-definite programming approach in the divide step.

1 Introduction

The study of evolution and the construction of phylogenetic (evolutionary) trees 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

[†] This research was supported by the Technion VPR-fund and by the Bernard Elkin Chair in Computer Science.
[‡] Supported by NSF Award–0331494.
[§] This reasearch was supported by National Institutes of Health Grant R01-HG02362-02.

R. Casadio and G. Myers (Eds.): WABI 2005, LNBI 3692, pp. 89–103, 2005.

based approaches for phylogenetic trees reconstruction. The goal behind the "tree of life" project is to 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.

Phylogeny reconstruction methods are broadly divided into *character-based* and *distance-based* methods. Distance based methods start by computing "evolutionary distances" between pairs of taxa. Then a tree with weighted edges whose pairwise tree distances approximate the evolutionary distances is sought, typically by some version of the *neighbor joining* clustering paradigm [SN87]. In contrast, character based methods work directly on character data. The best known and most widely used character-based methods are *maximum parsimony* [Fit81] and *maximum likelihood* [Fel81]. Maximum parsimony (MP) is a non-parametric combinatorial method, while maximum likelihood (ML) is a parametric statistical method. In MP, the tree sought is such that minimizes the total number of state changes on all edges for all the characters. MP was proved to be NP-hard already at 1982 by [FG82] while ML only recently by [CT05]. Nevertheless, for data comprise of up to few dozens of taxa, there exist good heuristics that are heavily used in practice.

The *supertree reconstruction problem* (or for short, the supertree problem) (to be defined rigorously later) is as follows: given a set of phylogenetic trees over overlapping non identical sets of taxa, find a tree over the union of the given taxa sets that *represents the best* the input subtrees. This output tree is denoted as the *supertree*. There are few criteria of how to measure the quality of the supertree. These criteria differ by the type of the input trees.

Phylogenetic trees are divided into *rooted* and *unrooted* trees. In unrooted trees, the decision problem of whether there exists a supertree that resolves all the input subtrees was shown to NP-hard by [Ste92]. 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.* , devised originally in the setting of relational data bases. Rooted trees have other attractive properties. In [SDB00] three required basic properties from a supertree method are listed:

- The method can be applied to any unordered set of input trees.
- if we rename all the species and then apply the method to the new input trees, we get the same old output tree under the renaming performed.
- if the input trees are consistent, the supertree returned should resolve all the input trees.

In the same work, it is shown that if the input tree are unrooted, there is no supertree method that can satisfy these three requirements simultaneously. However, when the setting is changed to a rooted setting, these requirements are achievable, in addition to polynomial running time of the method. They suggested that the rooted setting was perhaps superior to the unrooted one in the context of supertree construction. Based on these requirements, [SS00]

devised the min cut (MC) supertree method and showed it satisfies all of them. A later modification by [Pag02] to the (MC) algorithm, denoted the *modified min cut* (MMC) algorithm, guaranteed that the resulting supertree satisfied additional desirable properties. Page has made code available that implements this algorithm and maintains a server where one can run the MMC algorithm (at http://darwin.zoology.gla.ac.uk/ rpage/supertree/).

Briefly, both MC and MMC proceed recursively by finding a minimum cut in a graph built from the current set of triplets. The triplets in this minimum cut will then be discarded so that the remaining triplets can be partitioned into subtrees that are combined. The discarded triplets correspond to the unsatisfied ones. The algorithms are a bit narrow in their view in that they proceed cautiously in order to violate a minimum number of triplets at each step. At each step there are also triplets that are satisfied. They do not include this phenomena in their divide step.

In this paper, we extend the above approach and apply a more inclusive and less greedy criterion in the divide step. In short, instead of minimizing the number of triplets that are violated at every step, we aim at maximizing the ratio between satisfied to violated triplets. Unfortunately, maximizing this ratio is, itself, an NP-complete problem (by reduction from the sparsest cut problem for general demands). Thus, we develop a very fast heuristic that is motivated by semi-definite programming (inspired in part by work of Goemans and Williamson [GW95]) that works quite well for this application. This approach leads to practically much better results in terms of violated triplets without interfering with any of the properties guaranteed by these algorithms. Indeed, our methods are significantly better and significantly faster than implementations of MMC. Moreover, the implementation effort was quite modest.

We also compare our performance to a supertree method that is far slower but has significantly better performance for the triplet problem. This method proceeds by reducing an instance of a triplet problem to an instance of the matrix representation parsimony problem (MRP), and then using state of the art parsimony engines to solve the parsimony problem. This method previously produced far better results than the triplet based methods at the cost of much larger running times [ECB$^+$04]. Our methods also outperform these methods in terms of satisfying triplets. On one of our input distributions, MRP appears to do better on a fit measure involving the model tree, called maximum agreement subtree (MAST). On the other, we appear to do better. Also significantly, while for small sets of taxa we improve only modestly, as the number of taxa grows our advantage increases a bit. The computational requirements of the MRP approach, do not allow us to obtain performance numbers for MRP for very large number of taxa. We do report on results with our algorithms, however without comparing to other methods.

The use of semi-definite programming in the context of phylogenetic reconstruction is not new. Ben-Dor *et. al.* [BDCG$^+$98] have used semi-definite programming to construct a tree over a given set of quartets. However, we stress that our method and the one of [BDCG$^+$98] resemble only in the use of the

semi-definite programming technique. While [BDCG+98] use the embedding on the sphere once to resolve the whole tree and then use a neighbor joining type algorithm to reconstruct a complete tree, we use it at every step in the recursion only as an intermediate step to obtain an optimal partition of the taxa while the main algorithm is the divide and conquer Aho *et. al.* algorithm.

The remainder of this paper is organized as follows. In the next section we present the notations used and define the maximum triplet consistency method. In Section 3 we survey some existing methods for rooted supertrees and discuss some of their properties. Section 4 describes the pitfalls of current triplet based supertree methods, outlines our contribution, the algorithmic challenges involved and their resolutions. Section 5 describes experimental results comparing our method with two other representative methods on two types of synthetic data and also results on one instance of real data. We conclude in Section 6 with discussion and future research directions.

2 Preliminaries

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 $\mathcal{L}(T)$. Henceforth, we will identify the taxa set with the leaves they are mapped to. A tree T is said to be *rooted* if the set of edges E is directed and there is a single distinguished internal vertex r with in-degree 0. Let u and v be two vertices in a rooted tree T. We say that u is a *descendant* of v and v an *ancestor* of u, if there is a (directed) path from v to u. For $u, v \in V$, the *least common ancestor* of u and v, or $lca(u, v)$, is a vertex w that is an ancestor of both u and v and there is no descendant of w, w' that is also an ancestor of both u and v. From now on, we will restrict our attention to phylogenetic trees solely. Let T be a tree and $A \subseteq \mathcal{L}(T)$. We denote by $T|_A$ the tree induced by the sub set of leaves A where all internal vertices with degree two contracted. When T is rooted, the contraction is done at vertices with out-degree one. 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. In [BS00] it is shown that this requirement is equivalent to the condition that for every $u, v, w \in \mathcal{L}(T')$, $LCA(u, v)$ is a descendant of $LCA(u, w)$ in T if and only if $LCA(u, v)$ is a descendant of $LCA(u, w)$ in T'. This observation forms the basis to the triplet approach in supertree construction. 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 $T_i \in \mathcal{T}$. 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 *rooted triplet*, or for short a triplet, is a rooted tree over three leaves u, v, w. We write a rooted triplet over u, v, w as $u, v|w$ if $LCA(u, v)$ is a descendant of $LCA(u, w)$. The *triplet score* between a tree T_i and T^* is the sum of $u, v, w \in \mathcal{L}(T_i)$ such that $T_i|_{\{u,v,w\}} = T^*|_{\{u,v,w\}}$. The triplet score between $\mathcal{T}$ and T^* is

the sum of the triplet scores between T^* and every $T_i \in \mathcal{T}$. When $\mathcal{T}$ is a set of rooted triplets, this score is just the number of triplets satisfied by T^*. We refer to this problem as the *maximum triplet consistency* (MTC) problem.

Another score between a set of trees is the *maximum agreement sub tree* (MAST). This is defined as the largest set of leaves A common to all $T_i \in T$ such that $T_i|_A = T_j|_A$ for every $T_i, T_j \in \mathcal{T}$.

3 Supertree Methods

In this section we discuss competitive procedures for the supertree problem; the minimum cut methods and maximum parsimony and related methods.

3.1 Triplets Methods

We now describe minimum cut algorithms for triplet based reconstruction. The first algorithm solves the problem when all the triples are consistent and was developed in the context of relational databases by Aho *et. al.* [ASSU81]. Steel presented the algorithm for use in phylogenetics in [Ste92]. 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 $i, j|k \in \mathcal{T}:\ (i, j) \in E'$
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.

It is clear that the algorithm finds a tree T over $\mathcal{X}$ that satisfies all the input triplets if such a tree exists. The algorithm runs in time $O(|\mathcal{T}| \cdot n)$ and a later improvement by Henziger *et. al.* [HKW96] reduced it to $min\left\{O(|\mathcal{T}| \cdot n^{0.5}), O(|\mathcal{T}| + n^2 \log n)\right\}$.

As biological data suffers from noise it is very likely that $\mathcal{T}$ will be inconsistent, i.e. there will be no tree T that satisfies all the triplets in $\mathcal{T}$. Therefore it is desirable to find a tree that maximizes (minimizes) the number of satisfied

(violated) triplets. The Aho et. al. algorithm above will, thus, typically fail upon encountering such an inconsistency. Still, one can easily prove that it does not hurt to run it up to this point. That is, the following lemma can be proved.

Lemma 1. *Given an inconsistent set of triplets $\mathcal{T}$, but the connectivity graph G_C (constructed at step 3 of Aho* et. al. *algorithm) induced by $\mathcal{T}$ is not connected. Then, any optimal tree T^* for $\mathcal{T}$, will have a different subtree for every component in the connectivity graph.*

The min-cut algorithm of Semple an Steel [SS00] was the first to cope with this problem of inconsistent input triplets. Their algorithm handles rooted sub trees of arbitrary size and is centered around the following idea: Apply Aho *et. al.* algorithm as long as possible. if at a certain stage in the recursion of the algorithm, step 3 yields a connected graph (i.e a single connected component), perform the following modification to the algorithm:

- convert the connectivity graph G of step 3 into an edge weighted graph $G' = (V, E, w)$ where for $e = (i, j)$, $w(e) = |\{(i, j|k) \in \mathcal{T} : \ k \in X\}|$.
- Compute min-cut on the graph G'.
- Apply the Aho et.al. algorithm on the two subproblems induced by the two resulted components.

Although [SS00] did not prove any bound for the quality of their solution, they did claim that subtrees that are shared by all input subtrees, are resolved by the output supertree. An elegant extension of this idea was presented by Page [Pag02] to maintain all sub trees that are not in disagreement by any input subtree. The algorithm, denoted *modified min cut* (MMC) applies the min cut criterion but on a somewhat different graph that of [SS00].

3.2 Character Based Supertree Methods

We now describe a family of supertree methods that is based on solving a more general problem. The family of character based supertree methods contains these two methods: Matrix Representation using Parsimony (MRP) and Matrix Representation using Flipping (MRF). MRP [Bau92, Rag92] is the most widely used supertree method by practitioners. It has been found to have good performance [ECB+04]. 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. It is not confined to a rooted setting and hence loses some amount of its power when the input trees are rooted. However, to code for rooted trees, the following idea is employed: augment the taxa set with an artificial species s_r. Now let T_i be some input sub tree with root r_i and let e be an edge in T_i inducing the partition (A, B) over $\mathcal{L}(T_i)$. w.l.o.g assume that $T|_A$ contains r_i. Then $C(r_i) = 0$ and for all $s \in A$, $C(s) = 0$ and for $s \in B$, $C(s) = 1$. This is a reduction from the MTC problem to the MRP problem in the following sense.

Observation 1. *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 triplet coded for C is satisfied by T. Otherwise it is* 2.

Corollary 1. *The parsimony score of M (or alternatively, the MRP score) is m plus the number of triplets violated by T.*

Since the MTC is NP-complete, Corollary 1 implies that solving the MRP problem even for rooted triplets is NP-hard and therefore, some heuristics need to be employed. In practice, this leads to the somewhat confounding results that even when given a consistent set of triples MRP methods do not obtain a consistent solution as do the triplet methods.

The other character based supertree method is the Matrix Representation using Flipping (MRF)[CEFBS02, ECB+04]. MRF starts with the same matrix as MRP however, its objective function is somewhat different: It seeks for the minimum number of states flipping at the characters in order to make all characters compatible. In that case, the supertree is convex on all characters. This problem is somewhat a restricted version of the "big convex recoloring" problem introduced at [MS04], as the characters are restricted to two states only. By similar lines to Observation 1 the following observation is derived:

Observation 2. *Given a set of triplets $\mathcal{T}$ and let M be the partial matrix representing $\mathcal{T}$, the MRF score for M is exactly the MTC score for $\mathcal{T}$.*

It was found that MRP and MRF perform similarly in experiments [ECB+04]. Since MRF runs much slower, we compare our method just to MRP.

4 MAX Cut Tree Construction

In this section we describe our algorithm, MAX CUT triplets. Next we show by an example how it improves over the local algorithms MC and MMC. Our algorithm proceeds along the divide and conquer strategy of Aho *et. al.* algorithm identically to MC and MMC. However, it differs from MC and MMC by the action performed when a set of inconsistent triplets is encountered.

We first observe that the algorithm terminates since at any divide step, a nontrivial cut is produced, identically to the other algorithms. It is easy to verify the following observation:

Observation 3. *For every triplet $u, v|w$ such that a good edge (u, w) or (v, w) is in the cut but (u, v) is not, the triplet $u, v|w$ is satisfied by the algorithm.*

Therefore, it is plausible that the algorithm should maximize the ratio between good to bad edges in every divide step of the algorithm. We remark here that all triplet based algorithms do that maximization implicitly in cases when the triplets are consistent. At that time, the ratio is infinity.

MAX CUT triplets $(V, \mathcal{T})$

1. Let V be the set of taxa.
2. If $\mathcal{T} = \emptyset$ return a tree of depth 1 with all V sister taxa.
3. For every triplet $i, j|k \in \mathcal{T}$: $(i, j) \in E$
4. Let c be the number of connected components in $G = (V, E)$.
5. If $c = 1$
   - Denote the edges created in step 3 as *bad edges*
   - For every triplet $i, j|k \in \mathcal{T}$, augment two *good edges* (i, k) and (j, k) to E.
   - Find a cut $(C, \bar{C})$ in G such that the ratio of good edges versus bad edges in the cut is maximized.
6. create an internal vertex u.
7. For every connected component C_i in G,
   - $T_i \leftarrow$ MAX CUT triplets $(V(C_i), \{(i, j|k) \in \mathcal{T} : i, j, k \in V(C_i)\})$.
   - make T_i a child of u.
8. return T_u.

Heuristic for Optimizing the Ratio. Unfortunately, in general, finding a cut that maximizes the ratio of good edges to bad edges (that we use in step 5 of the algorithm) is NP-complete as well (for example, one can reduce from the max cut or sparsest cut problems.) Several semi-definite programming based approximation algorithms, however, have been suggested for related problems [GW95, ARV04]. Based on these approaches we developed the following heurstic.

The heuristic proceeds embedded vertices of the graph onto the surface of a 3 dimensional sphere by locally moving vertices to minimize the function

$$\sum_{\text{good edges} e=(i,j)} w(e)d(i, j) - \alpha \sum_{\text{bad edges} e=(i,j)} w(e)d(i, j), \tag{1}$$

for various values of the parameter α. Essentially, we search for an α where the value is less than 0. The intuition is that for a good cut, $(C, \overline{C})$ where the ratio of good edges to bad is at most α, mapping all the points in C to a vector v and all the points in $\overline{C}$ to $-v$, yields a negative value for the function 1. Finding the minimal such value, makes it where other higher ratio cuts don't have good embeddings. Thus, the minimal value of α where the embedding problem has a nontrivial solution is a lower bound on the cut ratio for all cuts. The embedding can be computed with semidefinite programming packages for embedding into n dimensinos. We, however, found that a local heuristic that embedded onto a 3 dimensional sphere sufficed for the results we present here. It is very fast, and it appears to be effective.

Once, we obtain the embedding, we produce a cut as follows, partitioned the vertices into two sets by selecting a random hyperplane to partition the points and producing the corresponding cut.

Finally, we used a version of the Fiduccia-Matheyes bisection improvement algorithm [KL70, FM82] to improve the resulting cut.

We remark that with a provably accurate optimization procedure that we could binary search for an optimal value of α will yield an optimal ratio cut. For our inputs, our heuristic worked quite well with only a few values of α.

We also remark that with a semi-definite solution, one could could actually get a bound on how far from optimal the cut procedure is from optimal. Since this is a side issue, we leave that for the future.

4.1 Example

We now illustrate our algorithm on an example and compare it to the local algorithms MC and MMC. Consider the set of triplets $\{(1,2|4),(2,3|5),(2,5|3),(4,5|2),(5,6|3)\}$. It can easily be shown that their connectivity graph contains a single component (depicted in Figure 1). Moreover, as every subset of leaves appears in at most one input tree, the weight of every edge in the connectivity graph is one. In this case, the local algorithms will apply the min cut criterion in order to partition the taxa and continue in the divide and conquer approach. However, a naive application of the min cut criterion is indifferent of where to partition the graph and pathological example is the removal of the edge $(1,2)$ violating the triplet $(1,2|4)$, then the edge $(4,5)$ violating the triplet $(4,5|2)$, then $(5,6)$ violating $(5,6|3)$ and eventually $(3,2)$ violating $(2,3|5)$. This gives a total of three violated triplets.

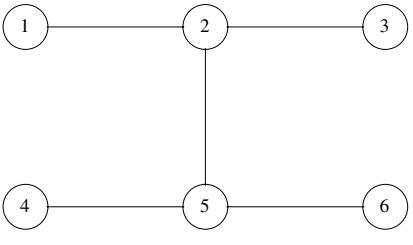

Fig. 1. A connectivity graph induced by the set of triplets $\{(1,2|4),(2,3|5),(2,5|3),(4,5|2),(5,6|3)\}$

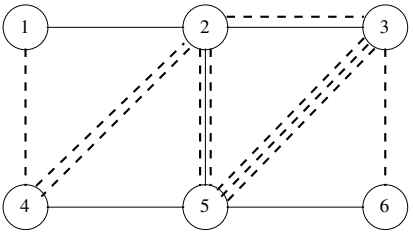

Fig. 2. The graph created by MAXCUT triplets algorithm for the set of triplets $\{(1,2|4),(2,3|5),(2,5|3),(4,5|2),(5,6|3)\}$. Good edges are drawn by dashed line.

In contrast, our algorithm constructs the graph depicted in Figure 2. It can easily be seen that the cut maximizing the ratio between good and bad edges is

the cut $(\{1,2,3\},\{4,5,6\})$. This cut violates the triplet $2,5|3$ but the rest of the triplets would be satisfied recursively by the algorithm.

5 Experiments and Discussion

In order to test our method, we conducted experiments that compared our method versus two previously mentioned methods. As we mentioned before, further results on these are reported on in [ECB+04]. Again, it appears that MRP is substantially more accurate than both MC and MMC but in the cost of a much longer running time. They also study a heuristic based on MRF as we previously mentioned. Here, we only test MRP and MMC versus our method since MRF was reported to have similar behavior to MRP with much longer running times.

We generated our triplets from some model tree. In particular, we generate triplets from a given model tree and output some percentage of incorrect triplets. The properties we wanted to measure are the accuracy of the supertree returned in terms of the number of triplets satisfied and the resemblance of that tree to the model tree.

We conducted two types of experiments differing by the way the species of any triplet were chosen.

1. **uniform:** The species are chosen according to a uniform distribution from the the species set.
2. **geometric:** Triplets over species with distance d were chosen with probability $\frac{1}{d}$. This introduces locality into the process of triplet generation.

5.1 Uniform Distribution Results

Our results in terms of triplet score, MAST score, and running times are reported in Table 1. We note that the gap in performances and running time between MMC and MRP in these experiments, is as reported in [ECB+04], while MRP and our procedure perform in a similar fashion. Indeed, the latter methods typically output a tree that is *better* than the model tree. We also observe our method is usually a bit better than MRP, and as the problem size grows our advantage increases. For example, for the largest problem where we could get scores for both, we obtain a score of 75.7 while MRP achieves 67.3. Moreover, the running of MRP (and even MMC) became prohibitive far before the limits of our methods. It appears that MRP outperforms our method in terms of MAST fit, however, as an evidence of the orthogonality of the two measures, it can be noted that even when the triplets were consistent (100% triplet fit for our method and MMC), still a higher MAST fit was obtained by MRP, although the tree it returned did not satisfy all the triplets.

Moreover, MXC is much faster than MRP and a fair bit faster than MMC. For example, MRP takes more than two hours and a half to solve a problem with four hundred triplets where MXC took 9 seconds. This is a factor of a thousand better. MMC also took more than twenty five minutes to solve a problem with

Table 1. Data from experiments of uniform distribution of triplet selection. "-" denotes that the problem took too long.

#taxa	#triplets	%correct	% Triplet fit			MAST fit			Running Time		
			MRP	MXC	MMC	MRP	MXC	MMC	MRP	MXC	MMC
50	400	80	81	81.5	40	17	14	8	13	1	4
50	1000	80	79	80.9	38.4	18	22	7	18	2	14
50	1000	100	99	100	100	27	26	26	12	1	6
100	100	50	96	100	100	11	8	8	74	1	1
100	400	70	70	77.7	52.7	14	12	8	152	3	17
100	1000	70	72.3	73.1	33.6	23	18	8	642	4	16
100	2000	50	49.6	54.5	34.1	19	18	8	402	7	63
150	100	70	96	100	100	14	6	6	290	1	1
150	1000	70	70	73.7	43.7	16	17	8	1712	4	35
150	1000	90	89.2	90.7	42.3	25	16	8	2628	5	29
150	2000	90	87.9	90.3	34.8	31	24	8	4790	17	108
200	400	70	81.2	81.5	65.7	17	10	9	9047	9	25
200	1000	70	67.3	75.7	44.1	15	17	9	8581	23	200
200	4000	50	-	54.2	34.5	-	25	200	-	53	546
400	10000	50	-	53.5	36.1	-	31	10	-	80	1603
400	50000	50	-	50.6	-	-	55	-	-	320	-
600	4000	50	-	62.4	-	-	17	-	-	38	-
600	50000	50	-	51	-	-	56	-	-	961	-
800	20000	50	-	53.5	-	-	34	-	-	132	-
1000	50000	50	-	51.4	-	-	49	-	-	339	-
2000	50000	50	-	53.5	-	-	-	-	-	383	-

ten thousand triplets and four hundred taxa where MXC took eighty seconds. It appears that MMC has some implementation problems as it crashed on twenty thousand triplets. The largest experiment we performed in this distribution was with fifty thousand triplets on two thousand taxa. The running time on this data was a bit more than six minutes (383 seconds). [1]

5.2 Geometric Distribution Results

In this type of experiment, we gave preferences to "close" over "far" triplets. That means that a triplet whose species are of distance d was chosen with probability $\frac{1}{d}$. We denote that as the geometric distribution. The 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, and therefore we "weigh" these distant triplets less. We believe this type of distribution is more realistic than the uniform distribution.

Table 2 depicts a sample of our experiments under the geometric distribution. It can be seen that both MRP and our method maintain the same superiority

[1] We note further that our code is hardly optimized in that while the divide step (the max ratio cut) is implemented in C, the recursive algorithm is written in perl with system calls to the max cut code. Thus, we are hopeful that this approach can work with even much larger datasets.

Table 2. Statistics on experiments of geometric distribution of triplet selection

#taxa	#triplets	%correct	% Triplet fit			MAST fit			Running Time		
			MRP	MXC	MMC	MRP	MXC	MMC	MRP	MXC	MMC
50	400	50	65.7	62	50.2	11	14	9	25	58	12
50	1000	70	65.3	70.3	43	21	29	13	44	7	26
50	2000	90	88.2	89.3	65.6	34	36	26	46	22	48
100	400	50	74.2	74	53	15	21	9	499	45	30
100	1000	50	62	63.4	38.1	17	24	12	806	30	78
100	1000	70	66.6	69.8	46.5	29	28	12	775	18	111
100	1000	90	82.7	85.5	0.49	39	45	15	411	10	53
150	1000	50	64.8	66.9	41.5	16	33	15	2564	65	43
150	2000	50	61.6	60	35	21	22	13	7553	1596	243
150	400	90	80.5	83.5	71.75	24	20	18	1114	4	13
200	1000	70	68.3	71.3	44.1	22	25	13	6317	11	66
200	400	90	83	86	73	19	19	15	3053	3	13
200	2000	90	83.9	86.35	38.65	84	93	15	10980	17	76
250	100	70	98	100	100	17	6	6	2036	2	3
250	2000	70	66.1	71.7	42	33	52	18	25114	29	92
250	1000	90	75.8	77.4	57.6	30	24	19	13626	38	217
300	2000	70	65	67.3	44.8	32	29	16	40176	38	302
300	400	90	88.5	97.2	91.7	25	22	20	27183	2	14
300	2000	90	71	74.9	49.8	37	46	19	35555	18	153

in terms of triplet fit score over MMC with average of 2% advantage to MXC over MRP. MXC is still much faster than MRP although in this distribution we needed to check for more values of α what increased the running time in some cases (e.g. the line with 150 species, 2000 triplets and 50% correctness). A somewhat interesting phenomenon, is that with this distribution, our method strictly outperforms MRP in terms of MAST score, in contrast to the uniform distribution where it appears MRP has some advantage. It is notable that even under this distribution MRP has better MAST score when the triplets are sparse, so MXC and MMC satisfy all the triplets but leave an unresolved tree, as implied by Aho *et. al.* algorithm (e.g. the line with 250 species, 100 triplets and 70% correctness). Perhaps this distribution of triplets along with the MXC algorithm is better for learning the true tree as compared to the uniform distribution of triplets.

5.3 Experiment on Real Data

Although our method was designed to handle input in a form of triplet trees, we wanted to test its behavior on real data. Real data comes normally in a form of trees over more than three species. When coming to apply a triplet based method, a separate task is to generate a "representative set" of triplets that will lead to the best construction of a tree. Since this task is beyond the scope of this paper, we used a rather naive way and generated *all* the triplets induced by *any* subtree. Of course this approach is biased and gives more representation to higher ancestral vertices over more recent ancestral vertices.

The real data we used for our experiment is composed of 158 source trees with 267 marsupial species from 107 published studies [CBEBP04]. The source trees were based on a wide range of data types, including molecular sequences, DNA hybridization, karyotypes, and immunological, morphological and behavioral data. The average number of species per tree is about 16.4. The number of triplets generated is 2,380,724. Although the true tree is assumed to be known and is found in **TREEBASE** [PSDW], we compared the results we obtained to either the set of triplets (triplet fit) or to the input subtrees (Robinson-Foulds).

In this experiment we used the Robinson-Foulds (RF) topological distance between two trees [RF81]. This distance equals the number of splits existing at exactly one tree. In addition, since MRP returns a fully resolved tree, whereas a triplet based method (seeks to) return the minimally resolved tree that satisfies most of the triplets, we measured a more MRP-favorable measure which is the number of common edges between the two trees. This is simply derived from the latter by $\frac{E_1+E_2-RF}{2}$, where E_i is the number of edges in tree i and RF is the Robinson-Foulds distance.

The results on this data using this method of triplets generation show strict advantage to MRP. The triplet fit achieved by MRP was 98.2% versus 96% by MXC. Both the RF distance and the number of common edges between the supertree and each of the subtrees were normalized by the number of species in the subtree. The final scores are the sum of the latter over all subtrees. For these measures, MRP achieved an average of 0.49 different edges per species and average of 0.95 common edges per species. MXC in turn, achieved inferior results of 0.65 different edges per species and 0.78 common edges per species.

6 Conclusion and Further Work

In this work we described a novel idea of using the semi-definite technique for the purpose of constructing triplet based supertrees. We introduced a less greedy partition criterion for the cases where the triplets are inconsistent. Moreover, we showed experimentally that although optimizing this criterion implies solving a NP-hard problem, a simple heuristic suffices to provide good performance. This in turn yields a very fast algorithm that outperforms in terms of running time even the theoretically efficient algorithms of mincut. Moreover, we showed that for the type of inputs we studied here, the performance of triplet based methods can exceed those of the heavier character based methods. These results pose semi-definite programming as a promising direction in the supertree field.

We want to emphasize that there are few major questions to be answered in this direction:

- While triplet based method try to maximize the number of satisfied triplets, this does not necessarily goes along with optimizing the other measures such as MAST or RF distance. It will be valuable to study the difference between these measures and to try to come with some conclusions for which really measures trees similarity.

- We saw that on real data, MRP is still superior over naively taking all triplets and trying to solve this problem. Moreover, MRP outperforms MXC even on that criterion, (although it solves an apparently different problem). Therefore, a more insightful approach for selecting which triplets to include is requested.
- Our experiments showed that actually, an optimal solution to the ratio cut problem is unnecessary. It would be of interest to explore the influence of using different ratios on the quality of the trees inferred by the method.

Acknowledgments

We would like to thank very much David Fernandez Baca and Duhong Chen for a lot of help with the many technicalities involved. We also thanks David Bryant, Benny Chor, Oliver Eulenstein, Arie Freund, Dick Karp, Rod Page, Mauricio Resende, Dror Rawitz, and Mike Steel for helpful discussions.

References

[ARV04] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Symposium on the Foundations of Computer Science*, 2004.

[ASSU81] 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 Journal of Computing*, 10(3):405–421, 1981.

[Bau92] B.R. Baum. Combining trees as a way of combining data sets for phylogenetic inference. *Taxon*, 41:3–10, 1992.

[BDCG+98] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg. Constructing phylogenies from quarbcgoptets: Elucidation of eutherian superordinal relationships. *Jour. of Comput. Biology*, 5(3):377–390, 1998.

[BS00] D.J. Bryant and M.A. Steel. Extension operations on sets of leaf-labelled trees. *Advances in Applied Mathematics*, 16(4):425–453, 2000.

[CBEBP04] M. Cardillo, O. R. P. Bininda-Emonds, E. Boakes, and A. Purvis. A species-level phylogenetic supertree of marsupials. *Journal of Zoology*, 264(1):11–31, 2004.

[CEFBS02] D. Chen, O. Eulenstein, D. Fernandez-Baca, and M. Sanderson. Supertrees by flipping. In *COCOON*, 2002.

[CT05] B. Chor and T. Tuller. Maximum likelihood of evolutionary trees is hard. In *RECOMB*, 2005.

[ECB+04] O. Eulenstein, D. Chen, J.G. Burleigh, D. Fernandez-Baca, and M.J. Sanderson. Performance of flip supertrees with a heuristic algorithm. *Systematic Biology*, 53(2):299–308, 2004.

[Fel81] J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.*, 17:368–376, 1981.

[FG82] L.R. Foulds and R.L Graham. The steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[Fit81] W. M. Fitch. A non-sequential method for constructing trees and hierarchical classifications. *Journal of Molecular Evolution*, 18(1):30–37, 1981.

[FM82] C.M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. In *Design Automation Conference*, pages 175–181, 1982.

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, November 1995.

[HKW96] M. R. Henzinger, V. King, and T. Warnow. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. In *SODA*, pages 333–340, 1996.

[KL70] B. W. Kernighan and S. Lin. An ecient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 29(2):291–307, 1970.

[MS04] S. Moran and S. Snir. Convex recoloring of strings and trees: Definitions, hardness results and algorithms. *submitted*, 2004.

[Pag02] R.D.M. Page. Modified mincut supertrees. In *R. Guigo and D Gusfield (Eds): WABI 2002, LNCS 2452*, 2002.

[PSDW] W. Piel, M. Sanderson, M. Donoghue, and M. Walsh. Treebase. http://www.treebase.org.

[Rag92] M.A. Ragan. Matrix representation in reconstructing phylogenetic-relationships among the eukaryotes. *Biosystems*, 28:47–55, 1992.

[RF81] D.R. Robinson and L.R Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[SDB00] M. Steel, A. Dress, and S. Boker. Simple but fundamental limitations on supertree and consensus tree methods. *Systematic Biology*, 49:363–368, 2000.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4, 1987.

[SS00] C. Semple and M. Steel. A supertree method for rooted trees. *Discrete Applied Mathematics*, 103:147–158, 2000.

[Ste92] M. Steel. The complexity of reconstructing trees from qualitative characters and subtress. *Journal of Classification*, 9(1):91–116, 1992.