Greedy Strict-Consensus Merger: A New Method to Combine Multiple Phylogenetic Trees

Usman Roshan* Bernard M.E. Moret† Tandy Warnow† Tiffani L. Williams*

Abstract

Large and comprehensive phylogenetic trees are desirable for studying macroevolutionary processes and for classification purposes. One approach for obtaining large phylogenies is to combine the topologies (or source trees) from previous phylogenetic studies. Tree reconstruction techniques that use the above methodology are known as supertree methods. In this paper, we develop a new supertree algorithm called Greedy Strict-Consensus Merger (GSCM) and compare it to Matrix Representation Parsimony (MRP), the most popular supertree method. We test the behavior of GSCM and MRP on biological datasets and examine their performance with respect to maximum parsimony scores and running time. Our results demonstrate that the GSCM method outperforms MRP with respect to both these criteria on all the datasets we examined.

Contact: usman@cs.utexas.edu
Keywords: disk covering methods, maximum parsimony, matrix parsimony, supertree, strict consensus merger

1 Introduction

Many phylogenetic reconstruction methods attempt to solve NP-hard optimization problems such as Maximum Parsimony (MP) and Maximum Likelihood (ML) (Felsenstein, 1981; Hillis *et al.*, 1996; Foulds & Graham, 1982; Steel, 1994), with the result that a biologically acceptable phylogenetic analysis can take years to complete on only a few hundred taxa (see (Chase *et al.*, 1993; Rice *et al.*, 1997) for a well known example). The computational requirements of large-scale phylogenetics have motivated systematists to develop alternative techniques for reconstructing evolutionary trees, such as so-called "supertree methods."

Supertree methods combine smaller, overlapping subtrees into a larger tree. They can thus use existing, published reconstructions on which the community agrees as well as combine the outcomes of reconstruction on decompositions of a large dataset. The most popular supertree method is Matrix Representation Parsimony (MRP) (Baum, 1992; Ragan, 1992), which has been used in a number of phylogenetic studies (Jones *et al.*, 2002; Liu *et al.*, 2001; Bininda-Emonds & Sanderson, 2001; Bininda-Emonds *et al.*, 1999; Purvis, 1995); Bininda-Emonds and colleagues examined the topological accuracy of reconstructions made with several variants of MRP in simulations (Bininda-Emonds & Sanderson, 2001), while Page compared it to a different method (based on mincuts) (Page, 2002). Beyond these three studies, little has been done to investigate the relative performance of supertree methods in an experimental setting.

Our study addresses the performance of Greedy Strict-Consensus Merger (GSCM) and MRP to assemble a single, supertree from subtrees. The mathematical optimization criterion Maximum Parsimony (MP) is used as our performance metric. We consider the following questions:

- How do supertree methods respond to subtrees constructed from different dataset decomposition techniques?
- Given a fixed collection of overlapping subtrees, what is the best method to assemble them into a single supertree?

*Department of Computer Science, University of Texas at Austin, usman,tandy@cs.utexas.edu
†Department of Computer Science, University of New Mexico, moret,tlw@cs.unm.edu

To answer the first question, we constructed source trees from nucleotide sequences using two decomposition strategies to create the overlapping subproblems. One decomposition technique is to randomly partition the dataset into overlapping subproblems. The other partitioning strategy is based on disk-covering methods (DCMs), which are geared towards reducing the evolutionary diameter within each subset. We address the second question by observing the performance of the supertree methods with respect to maximum parsimony scores and running time.

We compare GSCM and MRP on real datasets that range from 300 to more than 700 taxa. Our study shows that GSCM outperformed MRP in terms of running time and maximum parsimony scores.

The rest of the paper is organized as follows. We describe our supertree algorithms in Section 3. We explain our experimental design, datasets, and implementations in Section 4. Finally, we discuss our results in Section 5.

2 Supertree Reconstruction Algorithms

Supertree-building algorithms combine phylogenetic trees from multiple studies to produce a supertree. Formally, $L(T_i)$ is the set of leaves of a tree T_i. Let $\mathcal{T}$ represent the set of k source trees labeled $T_1, T_2, \ldots, T_k$. The complete set of taxa in $\mathcal{T}$ is $S = \cup_{i=1}^{k} L(T_i)$. A supertree method requires a set of trees $\mathcal{T}$ and outputs a tree $\mathcal{T}$ with the leaf set S. Although consensus techniques also work on topologies to produce a summarized phylogenetic tree, supertree reconstruction has the advantage of not requiring identical terminal taxa sets. Only overlapping sets are needed, which allows the supertree method to produce more comprehensive phylogenies from the smaller reconstructions. For an extensive review and critique of current supertree methods see (Bininda-Emonds *et al.*, 2002).

2.1 Greedy Strict-Consensus Merger (GSCM)

Our Greedy Strict-Consensus Merger (GSCM) supertree algorithm is an extension of SCM which is an essential component of the supertree construction algorithm of Disk Covering Methods (DCM) (Nakhleh *et al.*, 2001; Warnow *et al.*, 2001; Huson *et al.*, 1999a,b). The DCM supertree construction algorithm under certain conditions is guaranteed to construct the true tree (see (Huson *et al.*, 1999a)). Our previous studies (Moret *et al.*, 2003) show that the supertree construction algorithm of DCMs outperform the current standard supertree method, MRP, if it were used as the supertree component. Motivated by that study we have designed a supertree method which employs SCM and can be applied to any set of subtrees.

SCM The SCM method takes two trees T_1 and T_2 on possibly different leaf sets, identifies the set of leaves X that they share, and modifies T_1 and T_2 through a minimal set of edge contractions, so that they induce the same subtree on X—called the "backbone". Once T_1 and T_2 are modified in this way, they can be merged. The details of the SCM algorithm are given below.

- Set $X = L(T_1) \cap L(T_2)$. We call X the *backbone*, and it must satisfy $|X| \geq 3$. Otherwise, the merged tree will be unresolved.
- Compute the strict consensus T_X of T_1 and T_2, which are restricted to the leaf set X (i.e., $T_X = \text{Strict Consensus}(T_{1|X}, T_{2|X})$.
- Add the remaining subtrees from T_1 and T_2 into T_X while not violating any of the bipartitions in T_1 and T_2. Note that it is possible that some piece of T_1 or T_2 may attach onto the same edge of T_X. In this case, a *collision* occurs (see Figure 1).
- Return T_X.

Greedy SCM We adapt the SCM algorithm to handle arbitrary dataset decompositions by merging trees in a greedy fashion. Our GSCM algorithm merges a set of trees $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ and returns a tree leaf-labeled by $S = \cup_{i=1}^{k} L(T_i)$. For notation let $\mathit{resolution}(T)$ be the number of internal edges in T divided by $|L(T) - 3|$.

1. Set $\mathcal{S} = \{T_1, \ldots, T_k\}$
2. While $|\mathcal{S}| \geq 2$ do the following:

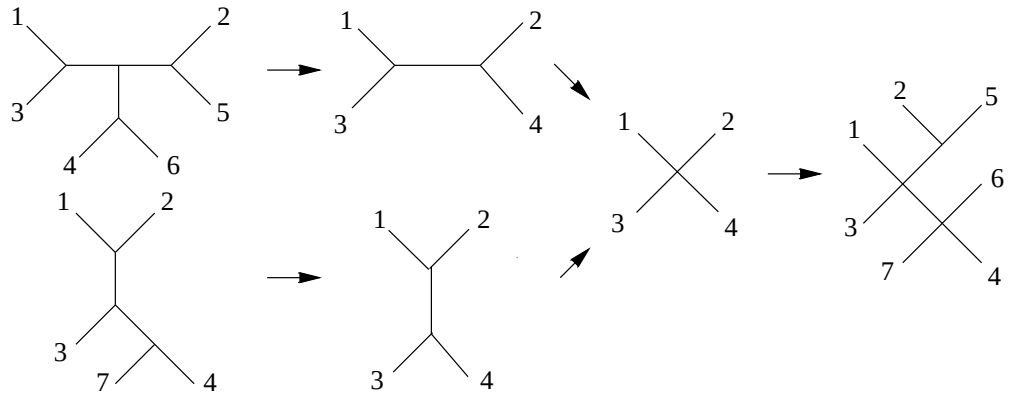

Figure 1: Merging two trees together, by first computing the strict consensus of the two trees restricted to the backbone, and adding remaining leaves so as not to violate bipartitions in the input trees. Note that the leaves 6 and 7 attach to the same edge which is a *collision*.

(a) Find trees T_i, T_j which maximize $\frac{\textit{resolution}(\text{SCM}(T_i,T_j))}{|L(T_i)\cup L(T_j)|}$, i.e. the normalized resolution of the Strict Consensus Merger of T_i and T_j. $\mathcal{S} = \mathcal{S} - \{T_i, T_j\}$.

(b) Set $T_i = \text{SCM}(T_i, T_j)$ where $\text{SCM}(T_i, T_j)$ is the Strict Consensus Merger of T_1 and T_2. Set $\mathcal{S} = \mathcal{S} \cup \{T_i\}$.

3. Return $\mathcal{S}$.

Since GSCM works with arbitrary dataset decompositions, there are no performance guarantees like there are for the DCM supertree construction algorithm, however we measure its performance by comparing it against MRP (see below) using simulation studies.

2.2 Matrix Representation Parsimony (MRP)

The MRP approach encodes a set $\mathcal{T}$ of k source trees labeled $T_1 \ldots T_k$ as binary characters with missing values (i.e., "partial binary characters") as follows. Consider a tree $T_i \in \mathcal{T}$ and let e be an arbitrary edge in T_i. The deletion of the edge e from T_i induces a bipartition π_e on $L(T_i)$ into sets A and B, where $L(T_i)$ is the leaf set of T_i. Let $S = \cup_{i=1}^{k} L(T_i)$. Now define a character c_e on all of S by extending π_e with $c_e(s) = 0$ for $s \in A$, $c_e(s) = 1$ for $s \in B$, and $c_e(s) =?$ for $s \notin L(T_i)$. The set $C(\mathcal{T}) = \{c_e \mid \exists T_i \in \mathcal{T},\ e \in E(T_i)\}$ is thus a set of partial binary characters that encodes all topological information of the trees in $\mathcal{T}$. Furthermore, if a supertree exists which exactly satisfies all the constraints in the set $\mathcal{T}$, then it will have optimal parsimony score with respect to $C(\mathcal{T})$. Hence, the supertree will be the maximum parsimony solution for S on this set of characters. Since MRP involves solving MP, it is an NP-hard problem and hill climbing heuristics are used in practice.

MRP implementation We used a medium speed MP search for MRP which uses 10 initial starting trees and 100 saved trees, however, this too can take a long time to finish, thus, we enforced a time limit on the search equal to the total time taken by Greedy-SCM followed by the refinement phase. We used the following PAUP*4.0 commands:

```
  set criterion=parsimony maxtrees=100 increase=no;
  hsearch start=stepwise addseq=random swap=tbr hold=1 nreps=10
timelimit=<total time taken to compute Greedy-SCM tree and refine it>;
```

3 Experimental Methodology

To analyze the behavior of GSCM and MRP supertree algorithms, our simulation procedure consists of the following four steps.

1. Decompose the dataset into smaller, overlapping subsets.

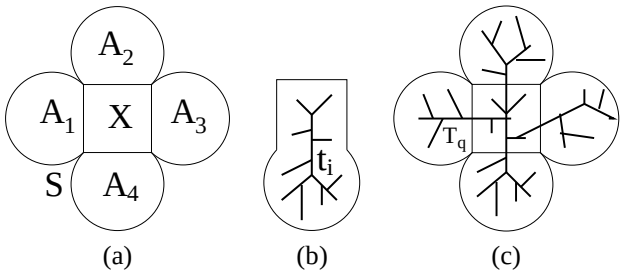

Figure 2: The three steps in Phase I of DCM2: (a) compute a clique separator X for S (relative to the triangulation of $G(d,q)$), producing subproblems $A_1 \cup X, A_2 \cup X, \ldots, A_r \cup X$; (b) compute a tree t_i for each subproblem $A_i \cup X$ using the chosen base method; and (c) merge the computed subtrees to obtain supertree T_q.

2. Construct phylogenetic trees on the subsets using the desired "base" phylogenetic reconstruction method.
3. Merge the subtrees into a single (not necessarily fully resolved) tree on the entire dataset.
4. Refine the resultant tree to produce a binary tree, and compare the MP scores.
5. Compare the running time of the supertree method, and the time to refine the supertree.

Initially, we begin with a set of nucleotide sequences that are partitioned into a set of overlapping subproblems based on two decomposition strategies—one decomposition is found using DCMs, and the other is a modification of the hierarchical clustering algorithm. The phylogenetic trees (or source trees) are constructed on these subsets using a maximum parsimony heuristic. The source trees are merged into a supertree by either the GSCM or MRP algorithm. If the supertree is not fully resolved, a heuristic search using maximum parsimony is performed to refine the merged supertree into a binary (i.e., bifurcating) tree. Even though the supertree methods are constructed on subtrees, we can compare their MP scores since we have the full set of sequences available, unlike in reality we may have *missing data* between the subtrees.

The MP score of a tree can be computed in polynomial time and is just the minimum number of times the sequences change across the tree. In order to measure the topological accuracy of a tree we need some definitions. The removal of an edge e in a tree T induces a *bipartition* on the leaves of T. Let $C(T)$ denote the set of bipartitions in tree T. If T is a model tree and T' is the tree obtained by a phylogenetic reconstruction method, then the false positive rate is defined as $\frac{|C(T')-C(T)|}{n-3}$ and the false negative rate is defined as $\frac{|C(T)-C(T')|}{n-3}$ where n is the number of leaves in tree T.

3.1 Data Decomposition

DCM2 decomposition DCMs are based on a divide-and-conquer strategy for phylogenetic reconstruction, based upon dividing a dataset into overlapping subsets, construction trees on these subsets, and then merging them into a tree on the full dataset. In (Huson *et al.*, 1999b) we described a DCM, called DCM2, for speeding up searches for Maximum Parsimony trees; we use that same DCM here. The input to DCM2 is a set $S = \{s_1, \ldots, s_n\}$ of n aligned biomolecular sequences, a matrix d containing an estimate of their interleaf distances, and a particular $q \in \{d_{ij}\}$. DCM2 then computes a threshold graph $G(d,q)$, as follows: the vertices of $G(d,q)$ are the taxa, $s_1, s_2, \ldots, s_n$, and the edges of $G(d,q)$ are those pairs (s_i, s_j) obeying $d_{i,j} \leq q$. A greedy heuristic is used to triangulate this graph so as to minimize the weight of the heaviest edge added; let G^* denote the triangulation of $G(d,q)$. We then compute a clique separator X of G^*, which minimizes the maximum size of any set defined as $X \cup A_i$, where A_i is one of the components of $G^* - X$ (Figure 2). Once we have this separator, we compute trees on each subproblem (the sets $X \cup A_i$).

DCM2 requires a particular threshold value, q, which influences the size of the subproblems examined (larger values tend to give larger subproblems for DCM2). We looked at five thresholds: d_0, the smallest threshold for which the threshold graph is connected, and d_2, d_4, d_6, and d_8 which are the 3rd, 5th, 7th, and 9th thresholds of evenly spaced values between d_0 and $d_9 = max\{d_{ij}\}$. Intuitively, higher thresholds means the subproblems are larger and the separator has a larger size also, thus larger overlap.

Random Decomposition We implement a method named RANDOM to decompose the dataset into random, overlapping subsets. This method requires three parameters: the number x of subproblems, the desired mean size y of each subproblem, and the desired size z of the setwise intersection of all the subsets.

We now describe how RANDOM(x,y,z) works. Let $n = |S|$, be the number of taxa to be distributed among the subsets. The x subsets are populated as follows. First, z taxa are randomly selected, and each of these z taxa are placed in each of the subsets. For each subset, we then randomly select an additional $(y-z)$ taxa from the remaining $(n-z)$ taxa. Finally, if any taxa have not been placed in any particular subset, we then add these taxa randomly to subsets. Note that this method does not guarantee that the subsets will have size exactly y; rather, every subset has size at least y. Furthermore, the cardinality of the setwise intersection is also guaranteed to be at least, but not necessarily exactly equal to, z.

In our experiments we varied subset sizes and fixed the *coverage* to be 2. The coverage can be thought of the number of subsets a taxon appears in, on the average. We chose the number of subproblems to be

$$\text{number of subproblems} = floor\left(\frac{\text{coverage}*\text{dataset-size}}{\text{average subproblem size}}\right)$$

The floor of a real number x is the largest integer that is less than or equal to x. We examined 5 values of average subproblem sizes, namely 10%, 30%, 50%, 70% and 90%.

3.2 Phylogenetic Base Method and OTR

We computed the subtrees on the subproblems using the RATCHET (Nixon, 1999) which is a maximum parsimony hill climbing heuristic that doubles the weight of 25% of randomly chosen sites once it reaches a local optimum, performs a hill climbing search in the *perturbed* space till it reaches a local optimum, sets the sites to their original weights and performs one more hill climbing search to completion. The result is the strict consensus of the best of the three trees found. We implemented the RATCHET in `PAUP*4.0`.

The Optimal Tree Refinement (OTR) problem is NP-hard, so we again use a fast heuristic search implemented in `PAUP*4.0`. We pass the unresolved supertree as a constraint tree to PAUP* and receive a binary tree refining the constraint tree, one that tends to optimize the parsimony score. We used the following PAUP*4.0 (Swofford, 1996) command:

```
constraints c1 (monophyly) = <constraint_tree>;
set criterion=parsimony maxtrees=1 increase=no;
pset collapse=no;
hsearch start=stepwise addseq=random swap=tbr hold=1 nreps=1
        constraints=c1 enforce=yes;
```

4 Experimental Design

4.1 Datasets and Implementation

Biological datasets We obtained six biological datasets (two DNA and four RNA) from various sources, listed below with the number of sequences, their lengths, and the maximum p-distance (normalized Hamming distance) between any two sequences in the set.

1. A set 328 ITS sequences (946 characters) from the flowering plant Asteracaeae obtained from the Gutell Lab at the Institute for Cellular and Molecular Biology, The University of Texas at Austin (max p-distance = 0.524).
2. A set of 388 aligned 16S rRNA sequences (2497 characters) for three different classes of Bacteria: Planctomycetes, Chlamydiae, and Verrucomicrobia (Maidak *et al.*, 2001) (max p-distance = 0.456).
3. A set of 500 aligned *rbcL* DNA sequences (1428 characters) also known as the Zilla dataset (Rice *et al.*, 1997) (max p-distance = 0.184).
4. A set of 567 "three gene: *rbcL*, *atpB*, and 18s" aligned sequences (2497 characters) of mostly angiosperms and and a carefully selected group of gymnosperms (Soltis *et al.*, 2000) (max p-distance = 0.15).

5. A set of 590 aligned small subunit Eukaryotic rRNA sequences (1962 characters) (Wuyts *et al.*, 2002) (max p-distance = 0.382).
6. A set of 778 aligned small subunit Mitochondrial rRNA sequences (1836 characters) (Wuyts *et al.*, 2002) (max p-distance = 0.656).

Since, both the decomposition methods require a distance matrix to compute the subproblems, we computed a distance matrix on each biological dataset using PAUP*4.0 by selecting the *ml* option (maximum likelihood) to estimate distances.

Implementation and Platforms Our data decomposition and supertree methods implementations are a combination of C++ (which uses LEDA 4.3) and Perl scripts which were written by us. Our experiments were run on modest Pentium 500 MHz machines under Debian Linux.

5 Experimental Results

Overview We began by comparing GSCM to MRP on RANDOM decompositions. We saw that GSCM trees were very poorly resolved (before OTR) on RANDOM decompositions. Thus, OTR would be doing most of the work. In that case, the comparison between GSCM and MRP becomes very difficult because most of the added edges (in the unresolved tree) would come from OTR. On the DCM2 decompositions, we saw that the GSCM trees were very well resolved.

A closer look at the RANDOM and DCM2 subproblems revealed that RANDOM subproblems had larger diameters compared to DCM2 decompositions. In practice biologists compute subtrees on sets of closely related sequences; thus, RANDOM decompositions serve as a poor model to capture a real life decomposition. We then proceeded to compare GSCM to MRP on DCM2 decompositions. There we saw that GSCM+OTR trees had better MP scores than MRP+OTR trees, and GSCM+OTR finished quicker than MRP+OTR.

5.1 Experiment 1: Comparison of GSCM and MRP resolutions on RANDOM and DCM2 decompositions

We computed RANDOM decompositions of five average subproblem sizes: $10\%, 30\%, 50\%, 70\%$, and 90% with the coverage parameter set to 2. We also computed DCM2 decompositions at 5 threshold values as described in Section 3.1. In our tables we list the DCM2 average subproblem sizes (rather than the threshold value) so that a comparison against RANDOM can be made easily. Note that the DCM2 average subproblem sizes sometimes start at large values; that only happens because we use the threshold parameter to compute DCM2 subproblems, and the average subproblem size for the first threshold value examined can be large. Subtrees on each decomposition were computed using the MP heuristic described in Section 3.2.

Table 1 compares the resolution of GSCM trees to MRP trees on RANDOM decompositions of the six biological datasets we studied. The resolution of a tree T is just the number of internal edges in T divided by $|L(T)-3|$ where $L(T)$ is the number of leaves in T ($|L(T)-3|$ would be the number of internal edges in an unrooted binary tree T).From Table 1 we can see that GSCM trees have poor resolution compared to the MRP ones. The resolution of both the trees improve as we move to larger subproblems. However, even at subproblems of average size 90%, the GSCM trees are all less than 75% resolved. Table 2 shows that on DCM2 decompositions, GSCM trees are much better resolved. There, most of the GSCM trees are more than 75% resolved.

To understand the difference in GSCM resolution on RANDOM and DCM2 decompositions, we proceed to compute the diameters of these decompositions. The *maximum diameter* of a subproblem s is the largest distance between all pairs of sequences $(i, j) \in s$, and the *average maximum diameter* is the the average of the maximum diameter of all the subproblems in a given decomposition.

Table 3 shows that RANDOM decompositions have larger diameters compared to DCM2 decompositions. A biologically realistic decomposition would have a small diameter since biologists construct trees on sets of closely related species. Thus, RANDOM decompositions do not capture a biologically realistic decomposition. We abandon

DataSet #1					
Avg RANDOM subproblem size	10%	30%	50%	70%	90%
GSCM	0.052	0.036	0.1	0.3	0.61
MRP	.41	0.6	0.84	0.89	0.96
DataSet #2					
Avg RANDOM subproblem size	10%	30%	50%	70%	90%
GSCM	0.025	0.01	0.072	0.21	0.45
MRP	0.21	0.57	0.81	0.86	0.92
DataSet #3					
Avg RANDOM subproblem size	10%	30%	50%	70%	90%
GSCM	0.002	0.008	0.016	0.19	0.46
MRP	0.39	0.67	0.94	0.96	0.96
DataSet #4					
Avg RANDOM subproblem size	10%	30%	50%	70%	90%
GSCM	0.001	0.003	0.028	0.22	0.60
MRP	0.42	0.78	0.96	0.98	0.98
DataSet #5					
Avg RANDOM subproblem size	10%	30%	50%	70%	90%
GSCM	0.001	0	0.02	0.18	0.44
MRP	0.39	0.80	0.89	0.93	0.96
DataSet #6					
Avg RANDOM subproblem size	10%	30%	50%	70%	90%
GSCM	0	0.001	0.01	0.07	0.36
MRP	0.33	0.83	0.94	0.96	0.98

Table 1: Degree of resolutions of GSCM and MRP on RANOM decompositions. GSCM returns poorly resolved trees (compared to MRP) leaving the OTR component to do all the work. RANDOM subproblems , however, have very large diameters which makes them biologically unrealistic.

DataSet #1					
Avg DCM2 subproblem size	78.0%	82.0%	86.0%	94.0%	95.0%
GSCM	0.75	0.68	0.79	0.74	0.79
MRP	0.95	0.95	0.94	0.94	0.86
DataSet #2					
Avg DCM2 subproblem size	54.0%	68.0%	79.0%	86.0%	93.0%
GSCM	0.81	0.77	0.76	0.68	0.74
MRP	0.72	0.79	0.8	0.76	0.84
DataSet #3					
Avg DCM2 subproblem size	85.0%	86.0%	90.0%	93.0%	97.0%
GSCM	0.74	0.76	0.78	0.78	0.81
MRP	0.93	0.95	0.85	0.86	0.87
DataSet #4					
Avg DCM2 subproblem size	54.0%	74.0%	98.0%	99.0%	99.0%
GSCM	0.98	0.89	0.82	0.88	0.86
MRP	0.98	0.98	0.97	0.92	0.9
DataSet #5					
Avg DCM2 subproblem size	68.0%	77.0%	84.0%	86.0%	95.0%
GSCM	0.75	0.76	0.79	0.79	0.76
MRP	0.91	0.92	0.88	0.86	0.84
DataSet #6					
Avg DCM2 subproblem size	88.0%	89.0%	95.0%	96.0%	97.0%
GSCM	0.64	0.64	0.61	0.59	0.62
MRP	0.94	0.92	0.93	0.92	0.90

Table 2: Degree of resolutions of GSCM and MRP on DCM2 decompositions. GSCM returns fairly resolved trees (compared to MRP). In this case we can make a fair comparison between GSCM+OTR to MRP+OTR

RANDOM and compare GSCM to MRP on DCM2 decompositions where the comparison can be made clearly, and where the decompositions are more biologically realistic than RANDOM ones.

Dataset#	1	2	3	4	5	6
DCM2 avg max diameter at avg subproblem size 96%	93.2%	98.6%	91.5%	93.3%	93.9%	78.2
RANDOM avg max diameter at avg subproblem size 90%	100%	100%	99.4%	99.7%	100%	95.5%

Table 3: Comparison of RANDOM and DCM2 subproblems. Note that the RANDOM subproblems have larger diameters. The maximum diameters are presented as a % of the complete dataset diameter

5.2 Experiment 2: Comparison of GSCM+OTR to MRP+OTR on DCM2 decompositions

We selected the four biological datasets with the highest average resolution of GSCM trees, and compared GSCM+OTR to MRP+OTR on those datasets. In particular, we selected datasets #1,#3,#4, and #5. We computed DCM2 decompositions on these datasets as described in Section 3.1, and constructed GSCM+OTR and MRP+OTR supertrees on each decomposition. We then compared the MP scores and the total running time of each method. The total running time is the time for constructing the supertree and applying OTR on it.

Figure 3 through 6 in the Appendix show that GSCM+OTR tree have better MP scores than MRP+OTR trees on all the datasets and all the average subproblem sizes. The improvement in MP scores by SCM+OTR is quite significant on all datasets and settings. GSCM+OTR returns trees which are many steps better than the MRP+OTR trees. Furthermore, GSCM+OTR is much faster than MRP+OTR. MRP takes a long time to finish if we were to let it run to completion.

6 Summary and Conclusions

We proposed a new supertree method and experimentally compare it to the standard popular supertree method used by biologists, MRP. We used random and DCM2-based dataset decompositions to obtain subtrees. We saw that random decompositions are not very biologically realistic and give subproblems with very large diameters; moreover, the comparison between GSCM+OTR and MRP+OTR cannot be made on random subproblems due to low resolution of GSCM trees before OTR.

On DCM2 decompositions, we see that the subproblems have smaller diameters (than RANDOM subproblems) and the GSCM trees are well-resolved (at least 75% resolution). Thus, DCM2 subproblems allow for a biologically realistic and fair comparison of GSCM+OTR to MRP+OTR. On DCM2 decompositions, GSCM+OTR has better MP scores than MRP+OTR and is much faster than MRP+OTR. Thus, for supertree construction GSCM may possibly return better supertrees in real studies.

In the future, we intend to explore the performance of GSCM on other types of dataset decompositions. It would also be interesting to apply GSCM to a real life supertree study and compare it to the MRP supertree.

7 Acknowledgments

This work was supported by the National Science Foundation under grants ACI 00-81404 (Moret), DEB 01-20709 (Moret and Warnow), EIA 01-13095 (Moret), EIA 01-13654 (Warnow), EIA 01-21377 (Moret), and EIA 01-21680 (Warnow), by the David and Lucile Packard Foundation (Warnow), and by an Alfred P. Sloan Foundation Postdoctoral Fellowship in Computational Molecular Biology, DOE grant DE-FG03-02ER63426 (Williams).