Performance of Supertree Methods on Various Dataset Decompositions

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

Abstract

Many large-scale phylogenetic reconstruction methods attempt to solve hard optimization problems (such as Maximum Parsimony (MP) and Maximum Likelihood (ML)), but they are limited severely by the number of taxa that they can handle in a reasonable time frame. A standard heuristic approach to this problem is the divide-and-conquer strategy: decompose the dataset into smaller subsets, solve the subsets (i.e., use MP or ML on each subset to obtain trees), then combine the solutions to the subsets into a solution to the original dataset. This last step, combining given trees into a single tree, is known as supertree construction in computational phylogenetics. The traditional application of supertree methods is to combine existing, published phylogenies into a single phylogeny. Here, we study supertree construction in the context of divide-and-conquer methods for large-scale tree reconstruction.

We study several divide-and-conquer approaches and experimentally demonstrate their advantage over Matrix Representation Parsimony (MRP), a traditional supertree technique, and over global heuristics such as the parsimony ratchet. On the ten large biological datasets under investigation, our study shows that the techniques used for dividing the dataset into subproblems as well as those used for merging them into a single solution strongly influence the quality of the supertree construction. In most cases, our merging technique—the Strict Consensus Merger (SCM)—outperforms MRP with respect to MP scores and running time. Divide-and-conquer techniques are also a highly competitive alternative to global heuristics such as the parsimony ratchet, although the relative performance depends upon characteristics of the dataset.

1 Introduction

Supertree methods combine smaller, overlapping subtrees into a larger tree. Their traditional application has been to combine existing, published phylogenies, on which the community agrees, into a tree leaf-labeled by the entire set of species. The most popular supertree method is Matrix Representation Parsimony (MRP) [1, 29], which

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

has been used in a number of phylogenetic studies [5, 6, 19, 21, 28]. Bininda-Emonds and colleagues [4, 6] have evaluated the behavior of several variants of MRP on small simulated datasets with respect to topological accuracy.

We study the application of supertree methods in a different context: as part of divide-and-conquer methods that can be used to solve difficult optimization problems such as Maximum Parsimony (MP) and Maximum Likelihood (ML) [11, 12, 16, 34]. These two problems are sufficiently hard that a biologically acceptable phylogenetic analysis can take a very long time (months, perhaps) to derive. Our conjecture, which we study in this paper, is that divide-and-conquer strategies can speed up searches for optimal trees under MP and ML.

A divide-and-conquer method for phylogeny reconstruction operates as follows. Given a dataset S of n sequences,

- Step 1: The set S is divided into overlapping subsets, $S_1, S_2, \ldots, S_p$.
- Step 2: A tree T_i is constructed on each subset S_i (e.g., by some heuristic search for MP or ML).
- Step 3: A supertree method is applied to the set of subtrees $\{T_i : i = 1, 2, \ldots, p\}$, in order to obtain a tree T on the full dataset.
- Step 4: If the tree T is not fully resolved (i.e., if it contains nodes of degree greater than three), a refinement technique is used to produce a binary tree refining T that optimizes the chosen criterion.

Supertree methods, therefore, are an integral part of a divide-and-conquer strategy, but the other three aspects of such a strategy also affect accuracy and speed. Our study addresses the following questions:

- Should the subtrees used in reconstruction be carefully selected in terms of the subsets they represent or can the subsets be arbitrary as long as some overlap exists among them?
- Given a fixed collection of overlapping subtrees, what is the best method to assemble them into a single supertree?
- How do divide-and-conquer methods fare when compared to "global" approaches, such as the heuristic searches for MP in PAUP*4.0b10 [36]?

To investigate the first two questions, we compare methods that differ explicitly in how they decompose the dataset and how they merge subtrees into a supertree. The family of Disk-Covering Methods (DCMs) [17, 18] plays a major role in addressing these questions. DCMs are metamethods for phylogenetic reconstruction that use a sophisticated dataset decomposition technique (based on cliques in a triangulated graph), that use a matched subtree merger (the Strict Consensus Merger or SCM), and that offer performance guarantees when used with particular base methods. We consider two variants in the DCM family, plus (as a control) random decompositions; these decompositions are coupled with MRP and SCM (only with DCM variants) to merge the resulting subtrees into a single supertree; finally, all combinations of methods are followed by a refinement phase. To ascertain whether divide-and-conquer approaches can outperform "global" approaches to solving MP or ML, we compare the performance of our DCM strategies with the parsimony ratchet (often cited as the best global MP heuristic).

1.1 Overview of Experimental Results

We compare these methods on ten biological datasets that range from 328 to 854 taxa, focusing on the question of how techniques used for dataset decomposition and supertree reconstruction impact the running time and the MP score of the result. We find that the DCM2+SCM method outperforms the other methods on all our datasets. The specific decomposition technique has a significant impact on the MP score of the resultant tree as well as on running time, with DCM2 clearly outperforming random decompositions. Furthermore, we obtain improved MP scores in all decomposition strategies (DCM and random) when the subproblems are large—an observation that impacts taxon-sampling strategies. The supertree method used to combine subtrees into a single tree on the full dataset is also very important. In general, except when the subproblems are very large and taxon coverage among the subproblems is high, SCM is faster than MRP and also produces better resolved trees. When MRP and SCM are followed by the same resolution technique in Step 4, SCM generally produces better MP scores than MRP. (The only exception was for DCM1-based decompositions, in which MRP produces very unresolved trees—compared to SCM, with the consequence that the refinement phase is able to produce better MP scores after MRP than after SCM—in effect, the refinement phase does all the reconstruction work, at a significant cost in running time.)

Our study demonstrates that the benefit of a divide-and-conquer technique depends on the properties of the dataset. When the dataset can be decomposed well by DCM2—into significantly smaller subproblems with good overlap, DCM2 provides a clear advantage (in running time or MP scores, as desired). We compared DCM2-based approaches with the parsimony ratchet—the best MP global heuristic in our experiments—on two biological datasets: the well-studied 500 *rbcL* DNA dataset and a set of 816 mitochondrial rRNA sequences. The *rbcL* dataset decomposes quite poorly and, indeed, our study shows that DCM2 provide no improvement over the parsimony ratchet; in contrast, the rRNA dataset decomposes very well and we find that DCM2 improves on the parsimony ratchet for this dataset. (Interestingly, DCM2-Ratchet, using the parsimony ratchet as a base method in a DCM2 decomposition, is almost as good as a global ratchet on the *rbcL* dataset, in spite of the extremely poor decomposition.)

1.2 Comparison with Previous Work

Bininda-Emonds and colleagues [4, 6] studied supertree reconstruction from an experimental point of view, focusing on the MRP method and using small simulated datasets. While we also study MRP, our focus is as much on decomposition as it is on supertree reconstruction and so we study several other methods; moreover, our testing uses biological datasets rather than simulated ones, thereby forcing us to use MP scores as our measure of accuracy (since the true trees for these datasets are not known); finally, we focus on large datasets (limited in this study to datasets below 1,000 taxa due to the dearth of larger published datasets), since these are the datasets where a divide-and-conquer methodology will have the largest impact.

Some of the earliest divide-and-conquer methods are quartet-based methods, such as Quartet Puzzling [35], Short Quartet methods [10], and Quartet Cleaning [2]. Quartet methods are at one extreme of divide-and-conquer methods, since they decompose the datasets into the smallest possible subsets for which nontrivial trees exist—subsets of just four taxa each. Quartet-based methods cannot profitably use either MRP or SCM (the two supertree reconstruction techniques we study here)—MRP is too expensive given the tiny trees and SCM will usually return a totally unresolved tree because too many quartets will be in error. In an earlier study [33], we compared various quartet-based methods and the fast and simple neighbor-joining (NJ) [31] method on simulated data. Quartet Puzzling, which merges quartet trees using a greedy heuristic, clearly dominated the other quartet-based methods, but was much slower and clearly less accurate than NJ. These results suggest that decompositions into tiny subsets is not profitable. Other divide-and-conquer methods include Compartmentalization [23], which is not fully described and so cannot be implemented, and a strategy used to analyze a biological dataset [26], where again the decomposition and merging steps are not well enough describe to enable one to implement and test the strategy.

2 Divide-and-Conquer Reconstruction Methods

Each of the divide-and-conquer methods uses four basic steps to construct a supertree from a given dataset:

- Step 1: Decompose the dataset into smaller, overlapping subsets.
- Step 2: Construct phylogenetic trees on the subsets using the desired "base" phylogenetic reconstruction method.
- Step 3: Merge the subtrees into a single (not necessarily fully resolved) tree on the entire dataset.
- Step 4: Refine the resultant tree to produce a binary tree.

Steps 2 and 4 are the same in all of our algorithms (except for our study of global heuristics versus divide-and-conquer methods in Section 5). We use a slow heuristic search for MP as the "base method" to construct the subtrees, but a fast heuristic search for MP to refine the merged supertree into a binary tree. Thus, our methods differ only in terms of how they perform Steps 1 and 3; Sections 2.2 and 2.3 describe the techniques used for data decomposition and subtree merging. A summary of all of the supertree methods used is given in Section 2.5.

2.1 Heuristic Searches for MP

Heuristic searches for MP trees form a basic part of our divide-and-conquer reconstructions in three places: using a base method on subproblems to construct subtrees, using MRP to merge subtrees into a supertree, and refining the resultant tree into a binary tree. PAUP*4.0b10 Heuristic Search (HS) [36] was used for these analyses since the datasets are too large (in the hundreds) for exact optimization. Experiments were performed on simulated data in order to determine the quality of the HS needed in each stage.

- Fast HS: A fast heuristic search in which we save only one tree, starting from one initial random sequence addition ordering. We used the PAUP*4.0b10 commands:
  ```
  set criterion=parsimony maxtrees=1 increase=no;
  hsearch start=stepwise addseq=random swap=tbr hold=1 nreps=1;
  ```
- Medium HS: Medium heuristic search with ten random sequence addition orderings and 100 saved trees. We used the PAUP*4.0b10 commands:
  ```
  set criterion=parsimony maxtrees=100 increase=no;
  hsearch start=stepwise addseq=random swap=tbr hold=1 nreps=10;
  contree all/ strict=yes;
  ```
- Slow HS: A slow heuristic search with 100 random sequence addition orderings and 1,000 saved trees. We used the PAUP*4.0b10 commands:
  ```
  set criterion=parsimony maxtrees=100 increase=no;
  hsearch start=stepwise addseq=random nreps=100 nchuck=1 chuckscore=1
  swap=tbr;
  set maxtrees=1000 increase=no;
  filter best=yes;
  hsearch start=current swap=tbr hold=1 nchuck=1000 timelimit=3600;
  contree all/ strict=yes;
  ```

We also used the parsimony ratchet [25], in a PAUP*4.0b10 implementation written by Bininda-Emonds [3]. The ratchet is a simple and effective heuristic for general optimizing search and works iteratively as follows:

1. Run Fast HS for MP.
2. Randomly select 25% of the sites, set their weights to 2 and run Fast HS on the perturbed data, starting with the tree from the previous search.
3. Reset the site weights to their original values and run Fast HS starting with the tree from the previous search.
4. Repeat steps two and three as desired.

2.2 Data Decomposition

2.2.1 DCM-based decomposition

Disk-Covering Methods (DCMs) [17, 18, 24, 37] are meta-methods for phylogenetic reconstruction: they operate in conjunction with a “base method” such as an MP heuristic or NJ. DCMs decompose the input set into smaller overlapping sets on which subtrees are computed using the specified base method. They have a dual goal: improved accuracy and better speed. Because the subsets have smaller diameter (maximum pairwise distance) than the original dataset, they are less likely to cause accuracy problems;

and because the possibly expensive base methods only have to solve small subsets, the overall algorithm runs faster. One goal can stressed at the expense of the other; thus there are several DCMs, each of which was designed for use with a particular base method.

Our first DCM, DCM1 [17], was designed for methods such as NJ, whose topological accuracy is negatively affected by large pairwise distances. DCM1 thus attempts to minimize the evolutionary diameter of each subproblem; in consequence, it produces many subproblems, each with small diameter, but does not control the overlap between the subproblems. Earlier studies we conducted (and confirmed here) showed that DCM1 does not work particularly well with heuristic MP as a base method. Therefore, we developed a second DCM, which we called DCM2 [18]. The main difference between DCM1 and DCM2 is that DCM2 produces a small number (two or three is typical in experiments presented here) of subproblems, all of which share one subset of taxa and are otherwise disjoint. Thus DCM2 tightly controls the overlap pattern, but does not directly attempt to control the diameter of each subset, thereby producing larger disks than would DCM1.

The input to both DCM1 and DCM2 is a set $S = \{s_1, \ldots, s_n\}$ of n taxa (typically, aligned biomolecular sequences), a matrix d containing an estimate of the pairwise distances between the taxa, and a *threshold*—a particular $q \in \{d_{ij}\}$. Both methods start by computing a *threshold graph*, $G(d,q)$, defined as follows:

- The vertices of $G(d,q)$ are the taxa, $s_1, s_2, \ldots, s_n$.
- The edges of $G(d,q)$ are those pairs (s_i, s_j) obeying $d_{i,j} \leq q$.

The graph is then minimally *triangulated*, i.e., edges are added to the graph until every cycle of length at least four has a chord (an edge connecting two nonconsecutive vertices on the cycle) [9, 14], while attempting to minimize the weight of the largest edge added. Obtaining an optimal triangulation of a graph is in general NP-hard [7], but threshold graphs are usually triangulated or close to it [17]—and our experience shows that even the simple greedy heuristic we use produces triangulations that do not have very long edges. We triangulate the threshold graph because triangulated graphs have many computationally useful properties, notably:

- they have a linear number of maximal cliques (cliques that cannot be increased by adding a vertex) and these cliques can be computed in polynomial time; and
- their minimal vertex separators (subgraphs whose removal breaks the graphs into disconnected pieces) are maximal cliques.

(These two problems are NP-hard for general graphs.) Thus, the next step in DCMs is to compute the maximal cliques. At this point, DCM1 is done and simply returns these cliques as the subproblems in the decomposition; these cliques have low diameter by construction. DCM2 scans through the cliques to find one clique X that minimizes $\max_i |X \cup A_i|$, where the A_i are the pieces into which the graph is broken upon removal of X; it then returns the subsets $X \cup A_i$ as the subproblems in the decomposition. Note that these subproblems have a unique common intersection, but that their diameter can be much larger (because of the addition of the separator X) than that of a subproblem generated by DCM1. Figure 1 illustrates the algorithm. We have proved that, as long

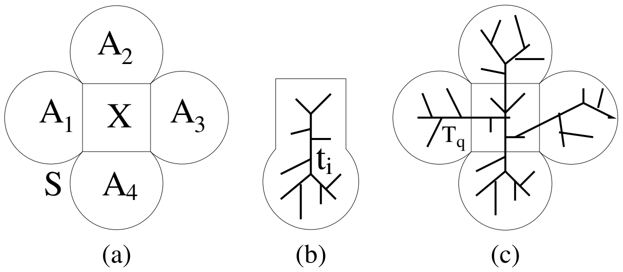

Figure 1: The three steps of Phase I in DCM2: (a) compute clique separator X for set S in threshold graph $G(d,q)$, producing subproblems $A_1 \cup X$, $A_2 \cup X$,..., $A_r \cup X$; (b) compute tree t_i for each subproblem $A_i \cup X$; and (c) merge computed subtrees to obtain tree T_q for set S.

as the subtrees are correctly inferred and the subproblems are large enough, the Strict Consensus Merger (SCM) technique applied to the subtrees will produce the true tree. These theorems have ramifications for both DCM1- and DCM2-based strategies, but especially for DCM1 combined with distance-based methods; for these combinations it is possible to prove nice theorems about the sequence length requirements of the resultant methods.

Our interest in this study is practical, however, rather than theoretical: we want to develop faster and more accurate algorithms that perform well in practice. We experiment with different decompositions in order to determine which ones produce the best empirical results, so that improved MP scores are obtained faster. The reason we pick a minimum triangulation is that we want to avoid grouping taxa that are evolutionarily distant; this is equivalent to avoiding the introduction of long edges in the triangulated threshold graph. In developing the threshold graph, however, we need to choose a threshold q. The smallest useful value for q is d_0, the smallest possible value for which the threshold graph $G(d,q)$ is connected; the largest possible value is simply $\max\{d_{ij}\}$ (but note that if we applied the algorithm to this largest value, then we would not obtain any decomposition into smaller subproblems, since the threshold graph would already be a clique). In our experiments we look at ten equally spaced values between d_0 and $\max\{d_{ij}\}$ and run all tests with two values: d_0 and d_4.

2.2.2 Random decomposition

As a control for DCM2, we also considered the effects of decomposing a dataset into random, overlapping subsets, using three parameters: the number x of subproblems, the desired minimum size y of each subproblem, and the desired minimum size z of the pairwise intersection of subsets. Let n 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 all of them are placed into 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 yet been placed in any particular subset, we add these taxa randomly to subsets.

The resulting decomposition mimics the structure of DCM2 in that it produces subsets with a common pairwise intersection.

2.3 Merging Subtrees

2.3.1 Matrix representation parsimony (MRP)

The MRP approach encodes a set $\mathcal{T}$ of trees as binary characters with missing values (i.e., "partial binary characters") and then applies some heuristic for maximum parsimony on the resultant set of sequences. Understanding how MRP works thus requires understanding the encoding and how maximum parsimony interprets partial binary characters.

Let S denote the full set of taxa and let T be one of the trees in the set $\mathcal{T}$—thus T has leaf set $S_0 \subset S$. Let e be an arbitrary edge in T. Deleting e from T partitions the leaves of T into two sets A and B. Now define a character c_e on all of S by setting

$$c_e(s) = \begin{cases} 0 & \text{if } s \in A \\ 1 & \text{if } s \in B \\ ? & \text{otherwise} \end{cases}$$

The set $C(\mathcal{T}) = \{c_e : \exists T \in \mathcal{T}, e \in E(T)\}$ is the MRP encoding of the set $\mathcal{T}$ of trees.

Given a set of sequences defined by partial binary characters and a candidate tree T on the set of sequences, all ?s are replaced by 0 or 1 in such a way as to minimize the total number of changes on (the parsimony score of) the tree. If every subtree in the MRP analysis is accurate (i.e., each subtree is identical topologically to the true tree induced on its set of leaves), then the true tree is one of the maximum parsimony trees. Hence, an exact solution to maximum parsimony will return the true tree as one of the solutions. (This observation follows from the fact that the true tree is a "perfect phylogeny" [7]—i.e., a homoplasy-free phylogeny, for the MRP-encoded set of sequences.)

For MRP, we used Slow HS. Since the search can identify more than one tree of lowest score, the strict consensus of the best trees found is returned—that is, the most resolved tree that is a common contraction of all of the best trees found.

2.3.2 Strict-consensus merger (SCM)

The Strict Consensus Merger (SCM) combines a set of trees into a single tree. The merging is done pairwise until only one tree is left. The specific order in which the trees are merged matters when the subtrees are defined by a DCM1 decomposition, but is irrelevant when the subtrees are defined by a DCM2 decomposition; hence, for DCM2 it suffices to describe how SCM operates on two trees. (For specifics on how SCM operates in a DCM1 analysis, see [17].)

Let $L(T)$ denote the set of leaves of T, $C(T)$ denote the set of bipartitions of T, and T_X, with $X \subseteq L(T)$, denote the tree obtained by restricting the leaf set of T to X and suppressing nodes of degree 2 (see Figure 2). SCM takes two trees T_1 and T_2 and returns a tree T_{12} on the leaf set $L(T_1) \cup L(T_2)$.

- Set $X = L(T_1) \cap L(T_2)$. X is the *backbone* and must satisfy $|X| \geq 3$.

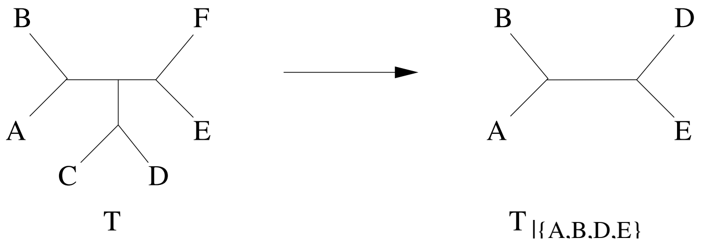

Figure 2: Tree T restricted to leaf set $\{A, B, D, E\}$

- Compute the strict consensus, T_X, of T_1 and T_2, each restricted to the leaf set X.
- Add the remaining taxa from T_1 and T_2 into T_X to form T, so as to preserve as much structure as possible. Some piece of each tree T_1 and T_2 may attach onto the same edge of T_X (causing a *collision*).

Figures 3 through 7 illustrate the SCM algorithm on both compatible and incompatible trees with and without collisions; in all figures, the backbone is highlighted with thick edges. In Figures 4, 6 and 7, there is a collision, i.e., an edge in the backbone to which both trees contribute pieces. The Strict Consensus Merger handles collisions in the following way. If an edge e of the backbone has a collision, then we subdivide the edge, producing a new node v_e, to which all contributions will be attached. In each subtree T contributing to this edge, we identify all pieces of T that should attach to that edge and attach them directly to v_e.

The Strict Consensus Merger of two trees is very similar to Gordon's Strict Consensus Supertree (SCS) [15]: the only difference is how collisions are handled: in case of collisions, the SCS tree is a strict contraction of the SCM tree, because it contracts additional edges located within pieces involved in the collision.

2.4 Optimal Tree Refinement (OTR)

Merging subtrees into supertrees using MRP or SCM can result in unresolved trees; all steps up to and including the merging step are perhaps best seen as attempts to identify the best-supported edges. Resolving the remaining polytomies (by adding edges) so as to minimize the parsimony score of the resulting tree is the NP-hard *Optimal Tree Refinement (OTR)* problem [8]. To "solve" it, we pass unresolved trees as constraint trees to PAUP*4.0b10 and use a fast MP heuristic search for a resolved tree, using the following command:

```
constraints c1 (monophyly) = <the unresolved tree which is used as constraint>;
set criterion=parsimony maxtrees=1 increase=no;
hsearch start=stepwise addseq=random swap=tbr hold=1 nreps=1
constraints=c1 enforce=yes;
```

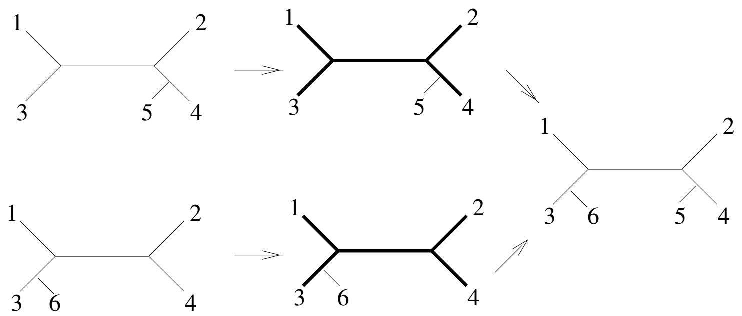

Figure 3: SCM on compatible trees without collision.

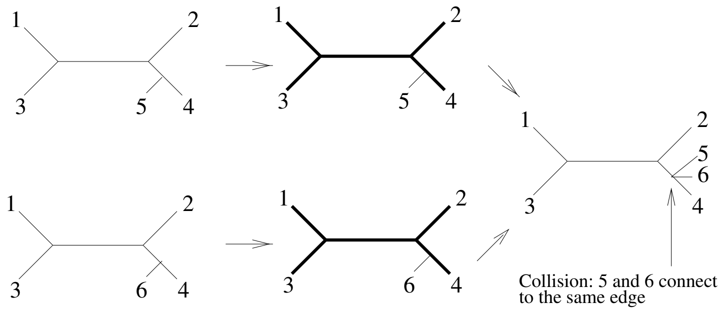

Figure 4: SCM on compatible trees with a collision.

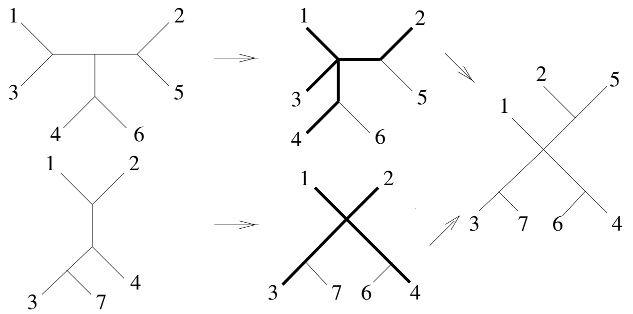

Figure 5: SCM on incompatible trees without collision.

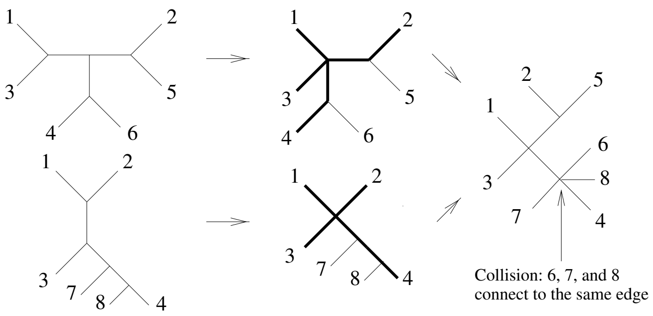

Figure 6: SCM on incompatible trees with a collision.

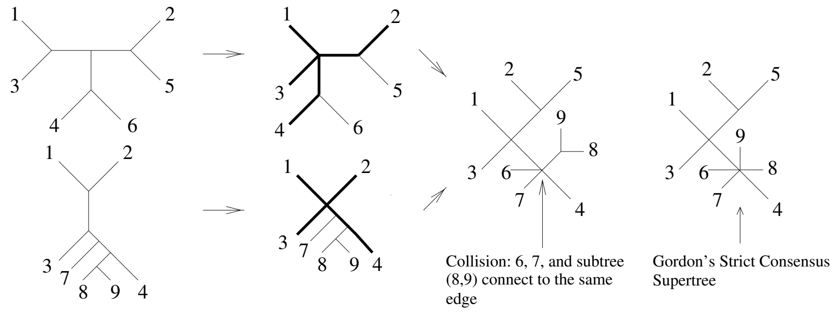

Figure 7: Handling collisions in the SCM and Gordon's Strict Consensus Supertree. The bipartition {{1,2,3,4,5,6,7},{8,9}} is present in the supertree under SCM, but not in Gordon's Strict Consensus Supertree.

2.5 Supertree Methods Studied

By varying the techniques used to obtain the dataset decomposition into subsets and the techniques used to merge subtrees into supertrees, we obtain many different divide-and-conquer methods. For each such method, we also have a choice of parameters. The methods we study are DCM1 and DCM2, each with a supertree construction phase of MRP or SCM, plus the random decomposition followed by MRP (SCM is only applicable to DCM decompositions). For DCM1, we use only threshold d_0, whereas for DCM2 we use both d_0 and d_4. All methods are followed by the refinement phase, described in the previous section.

3 Experimental Methodology

We ran two sets of experiments. The first set of experiments was designed to test two conjectures: (i) that careful decomposition of the dataset is crucial to the success of supertree methods and that the DCM methods offer such a careful decomposition; and (ii) that the strict consensus merger developed as part of DCM is superior to MRP as a supertree assembly tool. The second set of experiments was designed to test our conjecture that divide-and-conquer methods are a competitive alternative to global heuristics.

We used large biological datasets to test these conjectures. The advantage of biological datasets is that they offer data with all the biases and peculiarities that are so hard to produce in simulations and thus provide a better basis for prediction of future behavior on other biological data. Their disadvantages are that: (i) we cannot produce "tailored" biological datasets designed to test specific aspects of the reconstruction algorithms; and (ii) we cannot judge the outcomes on the basis of accuracy (because we do not know the "true" tree) and so must instead rely on substitute criteria, such as maximum parsimony scores or maximum likelihood scores. On balance, we chose biological datasets for two reasons: (i) we have already conducted large-scale simulation studies of the DCMs—it is those studies that led us to the conjectures listed above—and so already have strong supporting evidence for our conjectures from simulated data; and (ii) in contrast, no experimental study using large biological datasets has been conducted to date nor is there much known about the characteristics of such datasets.

Since our conjectures may hold in significant parts of the parameter space, but not everywhere, we study the effect of various parameter settings. We parameterize the decomposition in terms of subset sizes and mean coverage (where the mean coverage is the mean number of subsets in which a taxon appears). Of course, each taxon must appear in at least one subset, but reconstruction requires mean coverage greater than $1\times$ (otherwise we would obtain a forest and not a tree). We match the size and coverage characteristics of random decompositions to match our DCM decompositions and study the variation in parsimony scores as a function of subset sizes or coverage.

3.1 The Datasets

We obtained ten biological datasets (all biomolecular sequences) from various sources. Below we give a brief description of each dataset, noting the number of sequences, their lengths, and the maximum p-distance (normalized Hamming distance) between any two sequences in the set.

1. A set of 328 ITS RNA sequences (946 sites) from the flowering plant *Asteraceae* 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 439 aligned DNA sequences (2,461 sites) [13]; max p-distance = 0.649.
3. A set of 476 aligned DNA sequences (1,008 sites) [13]; max p-distance = 0.445.
4. A set of 500 aligned *rbcL* DNA sequences (1,398 sites) [30]; max p-distance = 0.184.
5. A set of 556 aligned 16S rRNA sequences (2,402 sites) for the Spirochaetes class of Bacteria [22]; max p-distance = 0.31.
6. A set of 567 "three gene: rbcL, atpB, and 18s" aligned DNA sequences (4,592 sites) [32]; max p-distance = 0.15.
7. A set of 590 aligned small subunit Archaea rRNA sequences (1,962 sites) [38]; max p-distance = 0.382.
8. A set of 695 aligned 16S rRNA sequences (2,550 sites) for the Cyanobacteria class of Bacteria [22]; max p-distance = 0.219.
9. A set of 816 aligned small subunit Mitochondrial rRNA sequences (1,253 sites) [38]; max p-distance = 0.46.
10. A set of 854 aligned DNA sequences (937 sites) [13]; max p-distance = 0.39.

Recall that the DCM-based approaches require a distance matrix to compute the threshold graph as the first step of its computation—and also that the distance matrix does not play any role in the phylogenetic reconstruction beyond this first step. We used the Kimura 2-parameter (plus Gamma) [20, 39] distance correction formula to compute a distance matrix for each dataset, using parameter values of $\kappa = 2$ and $\alpha = 1$ (the "default" values). We do not need the model to fit the data particularly well, since it affects only the choice of edges in the threshold graph; nevertheless, it is possible that a better distance correction (such as could be obtained using MODELTEST [27]) would yield better results. In other words, our results with the DCM-based approaches should be regarded as pessimistic—they establish a lower bound, but are subject to further improvement.

3.2 Implementation and Platforms

Our DCM implementations are a combination of C++ (which uses LEDA 4.3) and Perl scripts; they were originally written by Daniel Huson and further expanded by us. The random decomposition is also a combination of C++ and Perl scripts and was written by us. To run the MP heuristics used for solving the subproblems, for MRP, and for

OTR, we used PAUP*4.0b10 [36]. Our experiments were run on modest Pentium 500 MHz machines under Debian Linux.

For our running time analysis, we provide the running time (in seconds) of each of the four major steps separately, as follows:

- Decomposition: For the DCM-based methods this includes the running time for computing and triangulating the threshold graph and finding the subproblems. For the random methods it is the time to form the subproblems.
- Base method: This is the total running time for Slow HS on all the subproblems.
- Merge: This is the running time to merge the subtrees into a supertree using MRP or SCM.
- OTR: This is the cost of running Fast HS with the (unresolved) tree obtained in the previous step as a constraint tree in PAUP*4.0b10.

4 Results on Decompositions

4.1 Comparing Different DCMs

We began by examining the six different DCM-based approaches we defined: DCM1 using both SCM and MRP for supertree construction, but only using the threshold d_0, and DCM2 based upon both SCM and MRP, using thresholds d_0 and d_4. Figures 8 and 9 shows relative MP scores and running times on the ten datasets. The best method (in terms of MP scores) is consistently DCM2+SCM, at either d_0 or d_4; the other methods are not nearly as competitive. (MP scores that differ even by these small percentages are usually considered significant in phylogenetic analysis.) Furthermore, SCM is better than MRP at combining subtrees in nearly all cases—the only exception is DCM1 decompositions, but these decompositions are relatively poor and clearly not competitive, Running times show that MRP is far more expensive than SCM, and yet, as mentioned, is almost always dominated by SCM in terms of results. We will therefore focus on DCM2+SCM, since it is clearly the best performing divide-and-conquer strategy we tested. Our first task is to determine a suitable threshold. Figures 10 and 11 (using the data of Figures 8 and 9) show that DCM2+SCM(d_4) outperforms DCM2+SCM(d_0) on most of the ten datasets. This improvement in MP scores as we increase the threshold value is consistent with previous studies [18] and our recent simulation studies (not shown). Note that, as we increase the threshold, the number of subproblems decreases, so that, although the maximum subproblem size increases, the total time decreases. For $DCM2+SCM$, whether for threshold d_0 or d_4, the most costly aspect of the reconstruction is the time spent in the MP heuristics—in reconstructing trees on the subsets and, to a lesser extent, in the OTR phase; in contrast, the DCM and SCM phases are very fast. (These data are shown in the appendix.)

4.2 Comparing Random Decompositions

With random decompositions, we must use the MRP supertree method, since SCM is specifically designed for DCMs. Our goal here is to understand the effect of the

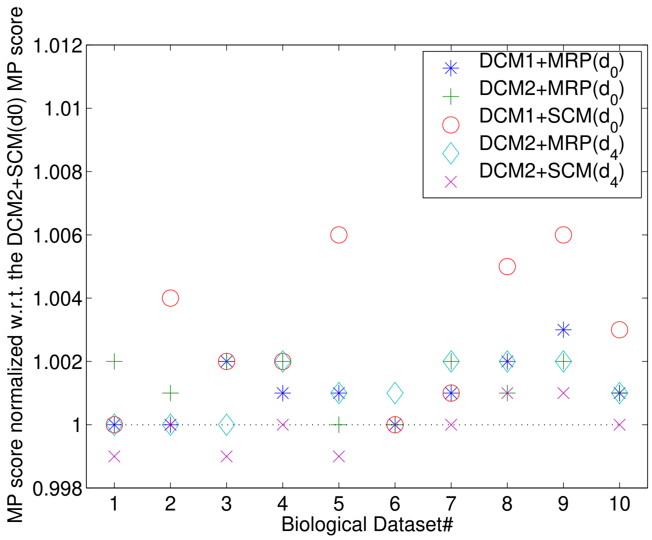

Figure 8: Comparison of the MP scores of DCM-based approaches on ten biological datasets, normalized with respect to the MP score of DCM2+SCM(d_0).

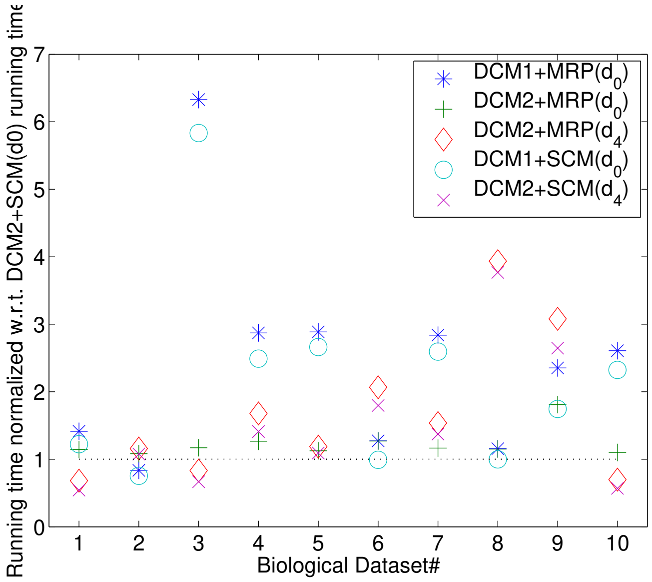

Figure 9: Comparison of the running times of DCM-based approaches on ten biological datasets, normalized with respect to the running time of DCM2+SCM(d_0).

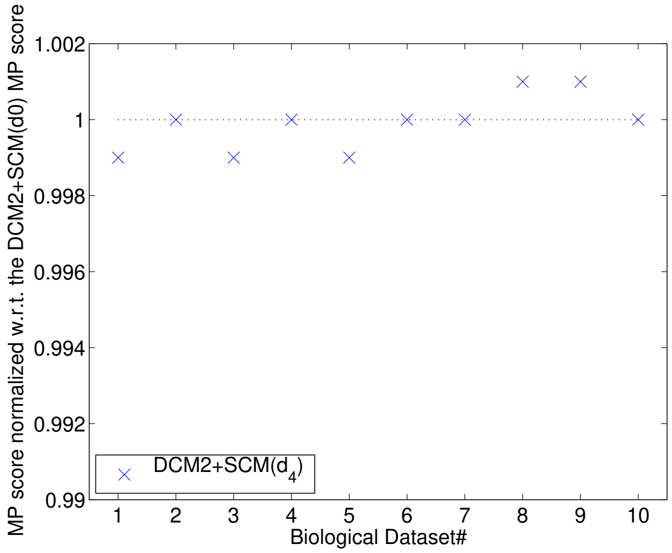

Figure 10: The ratio of the MP scores of DCM2+SCM(d_0) to those of DCM2+SCM(d_4) on ten biological datasets.

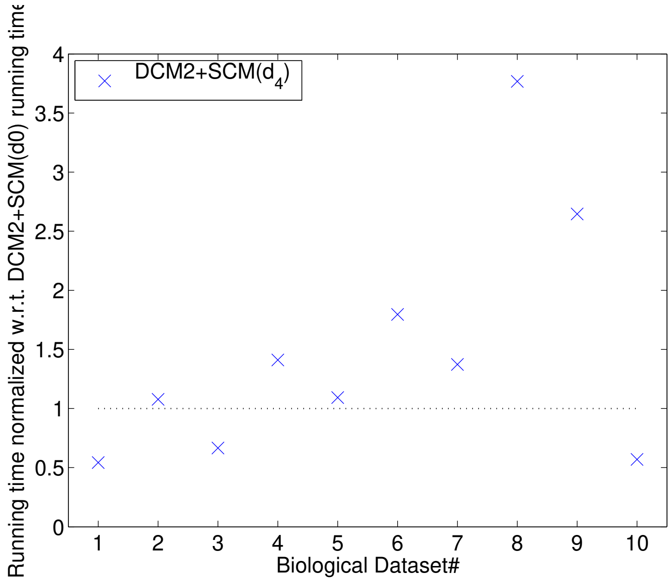

Figure 11: The ratio of the running times of DCM2+SCM(d_0) to those of DCM2+SCM(d_4) on ten biological datasets.

(random) decomposition, in particular, the size of subsets and the amount of coverage, on the quality of reconstructions.

We want the coverage (the average number of subsets in which a taxon appears) to run from $2\times$ to $5\times$ and the size of the subproblems to range from 10% to 90% of the dataset, so we we choose the number of subproblems to be

$$\text{number of subproblems} = \text{floor}\left(\frac{\text{coverage}\cdot\text{total size}}{\text{subproblem size}}\right)$$

So as not to bias the subset decomposition any further, we set the parameter z (the minimum overlap size) to 0 and let the pairwise overlap be induced through the number of subproblems and subproblem sizes as chosen above.

We solve the subproblems using Fast HS for MP (see Section 2) and run MRP using Medium HS for MP. We impose a time limit of 600 seconds on Medium HS. We examine 5 values of average subproblem sizes: 10%, 30%, 50%, 70% and 90% of the dataset. For each average subproblem size, we examine coverages of $2\times$, $3\times$, $4\times$ and $5\times$. Figures 18 through 27 in the Appendix show the results on each of the ten datasets. Shown are the ratios of the MP scores of MRP applied to random decompositions to the MP scores of DCM2+SCM(d_0), for various subset sizes (the horizontal axis, with the average subset size given as a percentage of the complete set) and coverages (the various curves). These figures indicate, unsurprisingly, that MRP applied to random decomposition does much better with larger subsets and somewhat better with increase coverage (as was also observed in [6]). Furthermore, as the subproblem sizes become larger, the MP scores of MRP on random decompositions slowly approach those of DCM2+SCM(d_0).

4.3 DCM vs. Random Decompositions

In our divide-and-conquer methods using random decompositions, we only use MRP as the supertree method, instead of also using SCM. The SCM method is designed for use on decompositions obtained on triangulated graphs, for which the order of mergers can be automated and has performance guarantees. However, with subproblems based on arbitrary decompositions, the order of mergers would need to be suitably defined, since it could have a large impact on the resultant supertree. Clearly, additional research would need to be done in order to establish a reliable use of SCM as a supertree method for arbitrarily defined sets of subtrees. We have begun this research, but have not yet completed it; for the purpose of this study, therefore, we restrict our use of SCM to DCM1 and DCM2 decompositions.

In this part of our study, we explore the relative performance of DCM2 decompositions to random decompositions. We run DCM2+SCM(d_0) only as a benchmark, but focus on DCM2+MRP(d_0), since we can ensure that MRP is applied to closely comparable decompositions. (We can set the three parameters for random decomposition so as to produce the same number of subsets as DCM2, with closely matched average subset sizes and coverage.) We use Slow HS for MP on the subproblems as well as for MRP and again report the average over 5 runs for the random decomposition. Tables 1 through 10 (in the Appendix) list the MP scores of the trees obtained by each method, while Figures 12 and 13 plot the ratios of MP scores (and running times) of

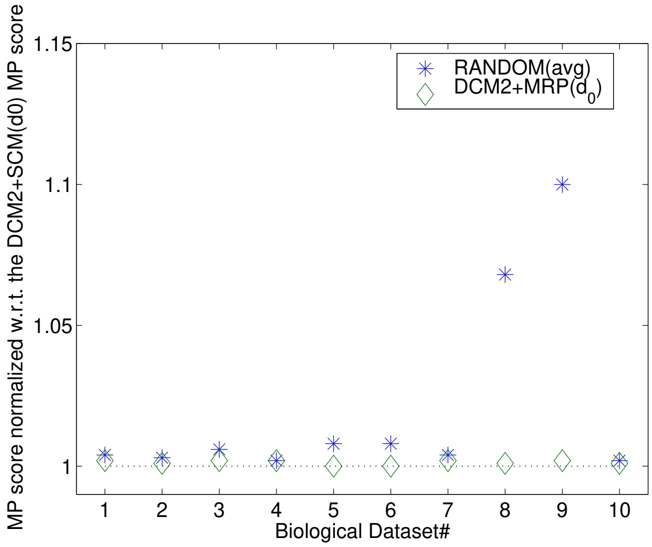

Figure 12: Comparison of MP scores of DCM-based methods and RANDOM (averaged over 5 runs) normalized with respect to the DCM2+SCM(d_0) MP scores on each of the ten biological datasets.

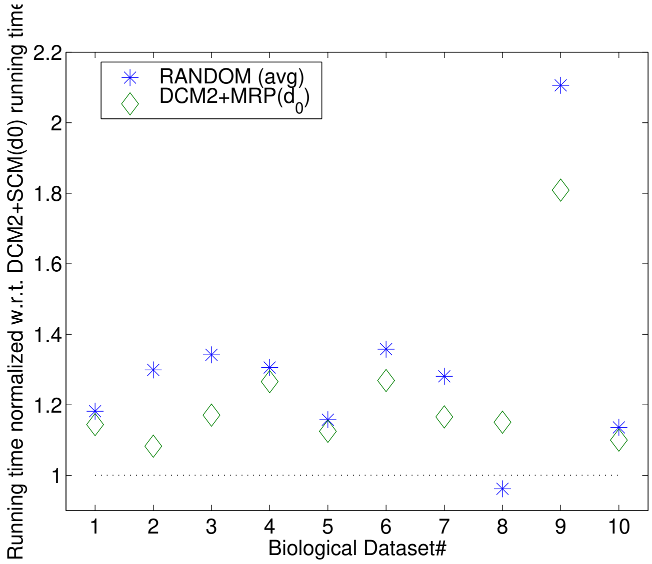

Figure 13: Comparison of running times of DCM-based methods and RANDOM (averaged over 5 runs) normalized with respect to the running time of DCM2+SCM(d_0), on each of the ten biological datasets.

DCM2+MRP and of MRP on random decompositions to the MP scores (and running times) of DCM2+SCM. The results clearly indicate that DCM2+SCM does much better, in terms of both MP scores and running times, than either DCM2+MRP or MRP on random decompositions, with this last doing worst of all. Thus DCM2 decompositions are better than random decompositions and SCM does a better job at assembling supertrees from such decompositions than MRP (scores and resolution are both better). In contrast, MRP is very slow (both DCM2+MRP and MRP on random decompositions run more slowly than DCM2+SCM because of the slow MRP phase)—on some datasets, the time difference is on the order of hours of computation, hours that could be used to conduct a more thorough parsimony search on the subtrees or in the OTR phase of DCM2+SCM. (We have not run such an equal-time comparison, but we expect that the gap in parsimony scores returned by DCM2+SCM and the other methods would be widened.)

5 Results on Global Heuristics

Our results suggest that a DCM2+SCM analysis is both faster and more accurate (in terms of MP scores) than the other divide-and-conquer methods studied. However, does DCM2+SCM compare well to a direct (global) heuristic approach? We expect that the DCM approach will prove better on those datasets that yield good decompositions (into a small number of substantially smaller datasets with good overlap), but need to ascertain how the DCM approach performs when decompositions are poor. We select two datasets, one for which the DCM2 decomposition is poor (dataset #4, the 500 *rbcL* dataset) and another for which DCM2 yields a good decomposition (dataset #9, the 816-taxon rRNA dataset).

We first explore various global heuristics to identify the method that performs best on the the 500 *rbcL* dataset—in this case the parsimony ratchet. We then use this heuristic both as a base method for DCM2+SCM (yielding a method we call DCM2-Ratchet, that also includes a final OTR phase) and as a global optimization heuristic and look at the progress of each method over the period of time needed by the global parsimony ratchet to find the best score (as of June 2003) for the dataset.

5.1 Local Improvement on the 500 *rbcL* Dataset

We compare local improvement heuristics as implemented in PAUP*4.0b10 on the 500 *rbcL* dataset. The heuristics we study are of the form **fast-k max-m** and are implemented using the following commands:

```
set criterion=parsimony maxtrees=k increase=no; hsearch start=stepwise addseq=random
nreps=k swap=tbr nchuck=1 chuckscore=1; set maxtrees=m increase=no; hsearch start=current
swap=tbr nchuck=m timelimit=<30 hours>;
```

We vary *k* and *m* to study the following versions:

- fast100-max1000
- fast1000-max1000
- fast10000-max1000

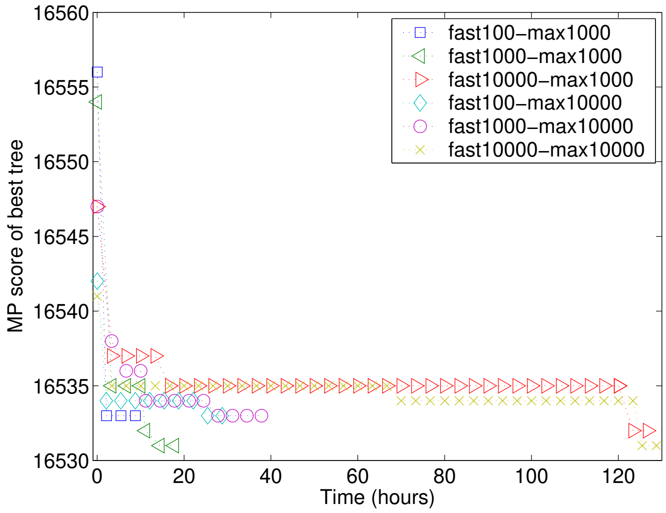

Figure 14: Comparison of six different hill-climbing heuristics on the *rbcL* dataset

- fast100-max10000
- fast1000-max10000
- fast10000-max10000

shows the MP score of the best tree found by each heuristic as a function of time—up to a time beyond which none of the heuristics finds better trees.

- fast1000-max1000 is clearly the best performing method in this study; it found the best tree of MP score 16,531 in the quickest time of approximately 14 hours.
- fast100-max1000 finds a tree of MP score 16,533 and is the fastest method.
- fast100-max10000 and fast1000-max10000 also find trees of MP scores 16,533; however, they take much longer than fast100-max1000.
- fast10000-max10000 and fast10000-max1000 are the two slowest heuristics; they find trees of scores 16,531 and 16,532 respectively after 125 hours.

We now compare 1,000 iterations of the parsimony ratchet (denoted by ratchet1000) against fast1000-max1000, the best of the PAUP* local improvement heuristics. Figure 15 shows that the parsimony ratchet finds trees with the same MP score as fast1000-max1000, but does so faster—at the 111th iteration only. This result, along with many statements in the literature about the observed speed of the parsimony ratchet, leads us to adopt it as our base method.

5.2 Global Ratchet vs. DCM2-Ratchet on the 500 *rbcL* Dataset

We explore several variants of DCM2-Ratchet, by restricting the number of iterations of the ratchet on the subproblems (to 25) and also by using the ratchet to resolve the

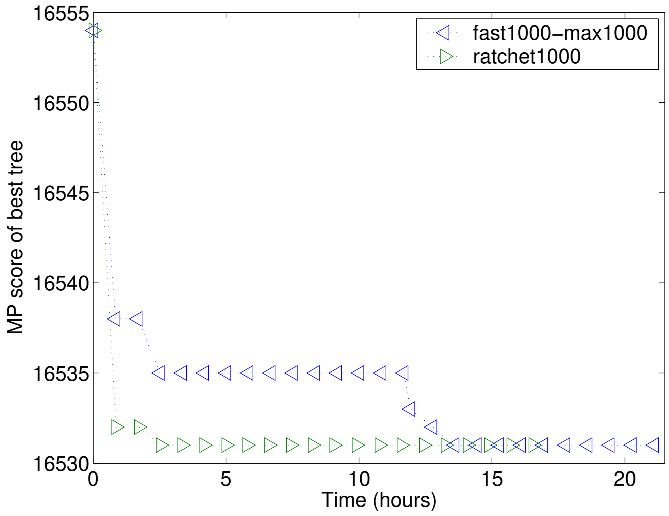

Figure 15: Comparison of ratchet1000 to fast1000-max1000 (on the *rbcL* dataset) shows that the parsimony ratchet finds the best known trees faster.

resultant tree (with iterations restricted to 10 and 25). The separator for the 500 *rbcL* dataset is very large—so large, in fact, that we elected to produce only two subproblems, rather than the three it naturally produces, by merging the two smaller subsets. Since ratchet1000 finds the best known trees on the 111th iteration (see Figure 15), we run 200 iterations of the ratchet for the global analysis. Each method is run 10 times and the average MP score at each time step is collected. Figure 16 shows that the DCM2-boosted variants of the ratchet are able to find trees almost as good as those found by the global ratchet, but not as fast. A closer look at the dataset decomposition shows that the DCM2 subproblems are very large at this threshold value (the average size is 90%!), which easily explains the slow running time.

5.3 Global Ratchet vs. DCM2-Ratchet on the 816 rRNA Dataset

On the 816 rRNA dataset, the DCM2 decomposition produced at threshold d_0 has a separator of size 3. Such a small separator makes it difficult for SCM to merge trees with sufficient accuracy, so we select instead a separator of size 36 (at threshold value d_0); this separator produces three subproblems of sizes 132, 270, and 486.

Since this is a larger dataset than the 500 *rbcL*, we use 500 iterations of ratchet (ratchet500) for the global analysis. The subproblems are again computed using 25 iterations of the ratchet and three different iteration counts are used for the OTR phase: ratchet5, ratchet10, and ratchet25. Each method is run 5 times and the average MP score at each time step is collected. The global ratchet version runs to completion in about 48 hours. Figure 17 shows that, within the first two hours, DCM2 finds better trees than the global ratchet. Moreover, when we run the global ratchet with the DCM2

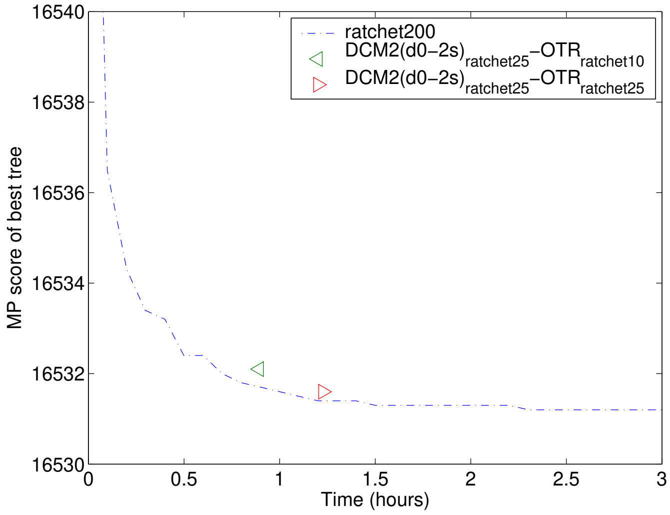

Figure 16: Comparison of DCM2-Ratchet to ratchet200 on Dataset #4 (averaged over 10 runs). The average subproblem size is 90% and the separator size is 79% of the problem size.

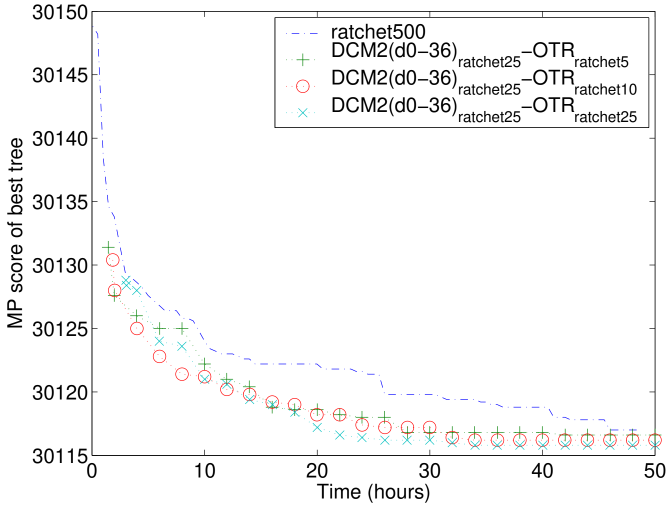

Figure 17: Comparison of DCM2-Ratchet to ratchet500 (best heuristic) on Dataset #9 (averaged over 5 runs). The average subproblem size is 36% (maximum is 60%) and the separator size is 4% of the problem size.

trees as starting trees, it finds better trees than the global version started from scratch. In particular, $DCM2_{ratchet25}$-$OTR_{ratchet25}$ finds trees with an MP score of 30,115 within approximately 26 hours, whereas the best trees found by the global ratchet (at the end of 48 hours) have an MP score of 30117,

6 Summary and Conclusions

We set out to explore the potential of divide-and-conquer methods to improve the speed and accuracy of maximum parsimony searches; in particular, we wanted to learn which decomposition strategies and what supertree assembly techniques work well in such approaches. Our study confirms that divide-and-conquer methods can speed up searches for optimal MP trees, but (unsurprisingly) only when the decompositions are good.

The specifics of the divide-and-conquer strategy make a large difference. We had already shown that quartet-based methods (an extreme form of divide-and-conquer) are not competitive. We now find that random decompositions are clearly inferior to carefully crafted ones (by the DCMs) and that the SCM approach to merging subtrees is both faster and more accurate than the traditional MRP approach. (Both approaches, however, usually require an "OTR" phase in which the supertree is refined into a binary tree, in order to get good results.)

The significance of this study is both enhanced and limited by our use of biological datasets: we ensure relevance, but can only conduct fairly simple tests—a simulation study is required to confirm our findings as well as discern more subtle effects. Other research suggested by this study includes: (i) how best to decompose datasets for which DCM2 does not produce a good decomposition? (ii) how long should the base methods be allowed to run on subproblems? (iii) what is the influence of the OTR phase on the entire process? and (iv) the DCM-SCM variants produce quite good trees fairly rapidly and thus can be used to initialize searches to good effect (as we show in this study), so how can we best make use of divide-and-conquer and global approaches? All of the work in this study concerns maximum parsimony, but divide-and-conquer methods, including the DCMs, are equally applicable to maximum likelihood—thus a study of DCM-ML approaches remains to be conducted. Finally, neither MP nor ML is the actual goal of phylogenetic reconstruction: both are surrogate optimization criteria for the real goal, which is topological accuracy (unmeasurable in absence of knowledge of the "truth"). Our simulations studies indicate that, while large decreases in parsimony scores or large increases in likelihood scores do translate into increase topological accuracy, small changes in these scores around near-optimal values have a nearly random effect on topological accuracy; this observation suggests that it may not be needed to spend additional days of computation to improve a score by 0.01% (since such an "improvement" could lead to a topologically worse tree) and thus that heuristic computations may be terminated earlier.

We conclude with a caveat: in reconstructing a very large tree, such as the Tree of Life with many millions of taxa, we may not have the luxury of choosing our decompositions—the data-gathering process may have made that choice for us, at least at the higher taxonomic levels. For instance, significant data may be missing for many taxa, so that it is not feasible to analyze all the sequences all at once. In such a case, the dataset decomposition will be given to us and thus will not be adjustable (except for the breaking of large clusters). We thus need a large-scale evaluation of supertree methods in their traditional use.

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 9985991 (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, U.S. Department of Energy DE-FG03-02ER63426 (Williams).

References

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

[2] V. Berry, T. Jiang, P. E. Kearney, M. Li, and T. Wareham. Quartet cleaning: Improved algorithms and simulation. In *Proc. 7th European Symposium on Algorithms (ESA 99)*, pages 313–324. Springer Verlag, 1999. Volume 1643 of LNCS.

[3] O. Bininda-Emonds. Ratchet implementation in paup*4.0b10, 2003. available from http://www.tierzucht.tum.de:8080/WWW/Homepages/Bininda-Emonds.

[4] O. R. P. Bininda-Emonds. MRP Supertree Construction in the Consensus Setting. In M. F. Janowitz, F. J. Lapointe, F. R. McMorris, B. Mirkin, and F. S. Roberts, editors, *Bioconsensus. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, volume 61, pages 231–242. American Mathematical Society, Providence, Rhode Island, 2003.

[5] O. R. P. Bininda-Emonds, J. L. Gittleman, and A. Purvis. Building trees by combining phylogenetic information: a complete phylogeny of the extant Carnivora (Mammalia). *Biological Reviews*, 74:143–175, 1999.

[6] O. R. P. Bininda-Emonds and M. J. Sanderson. Assessment of the accuracy of matrix representation with parsimony analysis supertree construction. *Syste. Biol*, 50:565–579, 2001.

[7] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Lecture notes in Computer Science 623*, pages 273–283. Springer-Verlag, 1992.

[8] M. L. Bonet, M. Steel, T. Warnow, and S. Yooseph. Better methods for solving parsimony and compatibility. *Journal of Computational Biology*, 5(3):391–408, 1998.

[9] P. Buneman. A characterization of rigid circuit graphs. *Discrete Mathematics*, 9:205–212, 1974.

[10] P. L. Erdős, M. Steel, L. Székély, and T. Warnow. A few logs suffice to build almost all trees– I. *Random Structures and Algorithms*, 14:153–184, 1997.

[11] J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

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

[13] P. Goloboff. Analyzing large data sets in reasonable times: solution for composite optima. *Cladistics*, 15:415–428, 1999.

[14] M. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press Inc, 1980.

[15] A. D. Gordon. Consensus supertrees: the synthesis of rooted trees containing overlapping sets of leaves. *Journal of Classification*, 3:335–348, 1986.

[16] D. Hillis, C. Moritz, and B. Mable. *Molecular Systematics*. Sinauer Pub., Boston, 1996.

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

[18] D. Huson, L. Vawter, and T. Warnow. Solving large scale phylogenetic problems using DCM2. In *ISMB 99*, pages 118–129, 1999.

[19] K. E. Jones, A. Purvis, A. MacLarnon, O. R. P. Bininda-Emonds, and N. B. Simmons. A phylogenetic supertree of the bats (mammalia: Chiroptera). *Biological Reviews*, 77(2):223–259, 2002.

[20] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120, 1980.

[21] F.-G. R. Liu, M. M. Miyamoto, N. P. Freire, P. Ong, and M. Tennant. Molecular and morphological supertrees for eutherian (placental) mammals. *Science*, 291(5509):1786–1789, 2001.

[22] B. Maidak, J. Cole, T. Lilburn, C. P. Jr, P. Saxman, R. Farris, G. Garrity, G. Olsen, T. Schmidt, and J. Tiedje. The RDP-II (Ribosomal Database Project). *Nucleic Acids Res*, 29(1):173–4, 2001.

[23] B. D. Mishler. Cladistic analysis of molecular and morphological data. *American Journal of Physical Anthropology*, 94:143–156, 1994.

[24] L. Nakhleh, U. Roshan, K. St. John, J. Sun, and T. Warnow. Designing fast converging phylogenetic methods. In *Proc. 9th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB01)*, volume 17 of *Bioinformatics*, pages S190–S198. Oxford U. Press, 2001.

[25] K. C. Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:407–414, 1999.

[26] G. J. Olsen, C. R. Woese, and R. Overbeek. The winds of (evolutionary) change: breathing new life into microbiology. *Journal of Bacteriology*, 176:1–6, 1994.

[27] D. Posada and K. A. Crandall. Modeltest: testing the model of dna substitution. *Bioinformatics*, 14(9):817–818, 1998.

[28] A. Purvis. A composite estimate of primate phylogeny. *Philosophical Transactions of the Royal Society of London Series B*, 348:405–421, 1995.

[29] M. Ragan. Phylogenetic inference based on matrix representation of trees. *Mol. Phylogenet. Evol.*, 1:53–58, 1992.

[30] K. Rice, M. Donoghue, and R. Olmstead. Analyzing large datasets: *rbcL* 500 revisited. *Systematic Biology*, 46(3):554–563, 1997.

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

[32] D. E. Soltis, P. S. Soltis, M. W. Chase, M. E. Mort, D. C. Albach, M. Zanis, V. Savolainen, W. H. Hahn, S. B. Hoot, M. F. Fay, M. Axtell, S. M. Swensen, L. M. Prince, W. J. Kress, K. C. Nixon, and J. S. Farris. Angiosperm phylogeny inferred from 18s rDNA, *rbcL*, and *atpB* sequences. *Botanical Journal of the Linnean Society*, 133:381–461, 2000.

[33] K. St. John, T. Warnow, B. Moret, and L. Vawter. Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. In *Proc. 12th Ann. Symp. Discrete Algorithms (SODA'01)*, pages 196–205. SIAM Press, 2001.

[34] M. A. Steel. The maximum likelihood point for a phylogenetic tree is not unique. *Systematic Biology*, 43(4):560–564, 1994.

[35] K. Strimmer and A. von Haeseler. Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution*, 13(7):964–969, 1996.

[36] D. L. Swofford. PAUP*: Phylogenetic analysis using parsimony (and other methods), 2002. Sinauer Associates, Underland, Massachusetts, Version 4.0.

[37] T. Warnow, B. Moret, and K. St. John. Absolute convergence: True trees from short sequences. In *Proc. 12th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA'01)*, pages 186–195. SIAM Press, 2001.

[38] J. Wuyts, Y. V. de Peer, T. Winkelmans, and R. D. Wachter. The European database on small subunit ribosomal RNA. *Nucleic Acids Res*, 30:183–185, 2002.

[39] Z. Yang. Maximum likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Mol. Biol. Evol.*, 10:1396–1401, 1993.

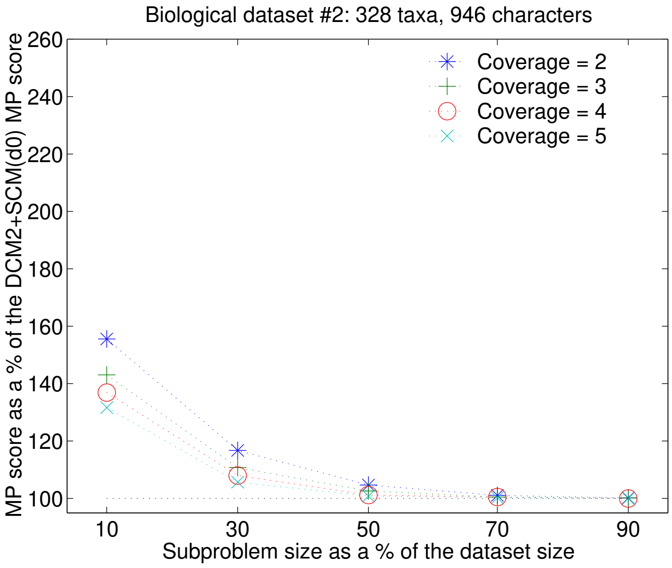

Figure 18: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #1 of 328 taxa and 946 sites

Appendix: Detailed Experimental Results

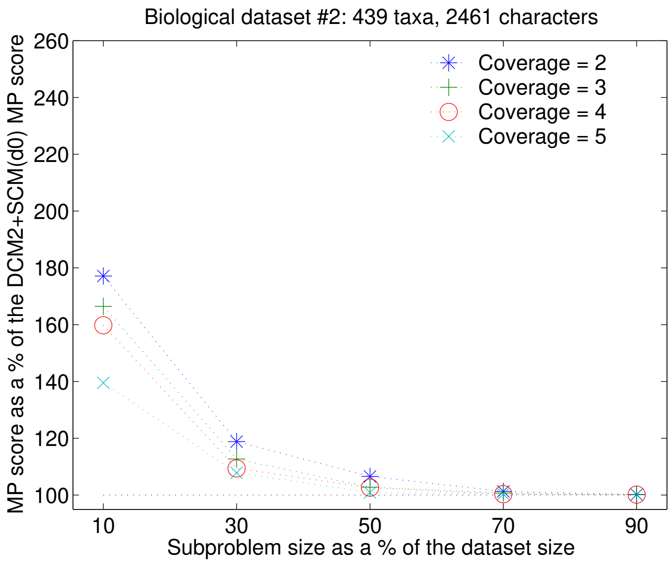

Figure 19: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #2 of 439 taxa and 2,461 sites

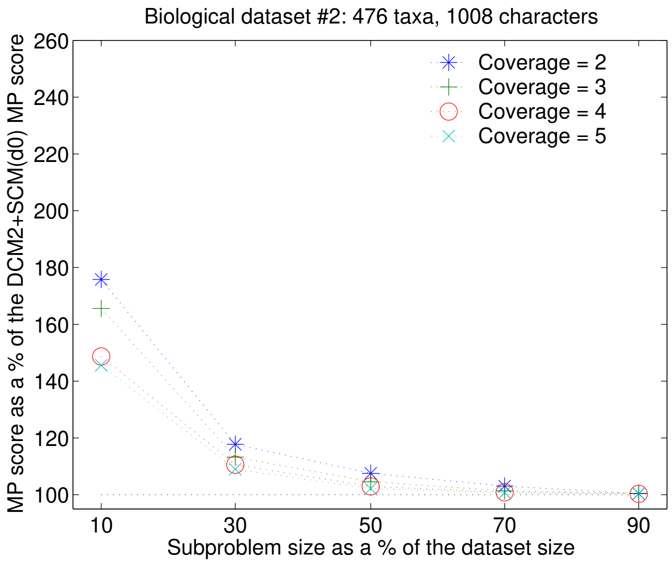

Figure 20: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #3 of 476 taxa and 1,008 sites

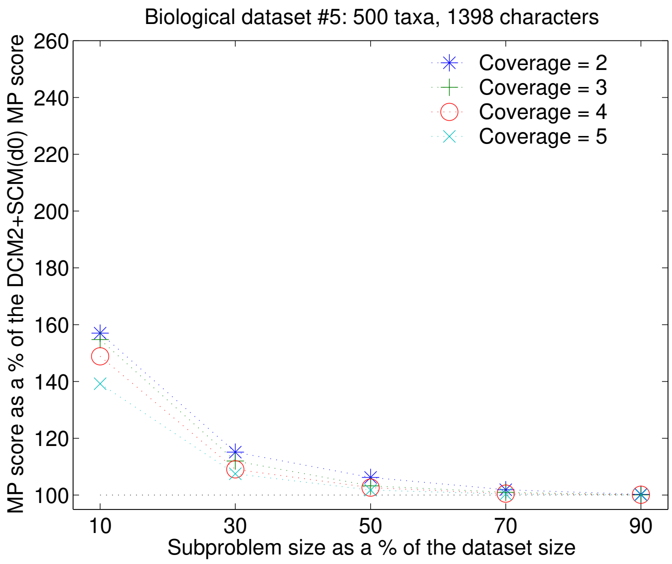

Figure 21: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #4 of 500 taxa and 1,398 sites

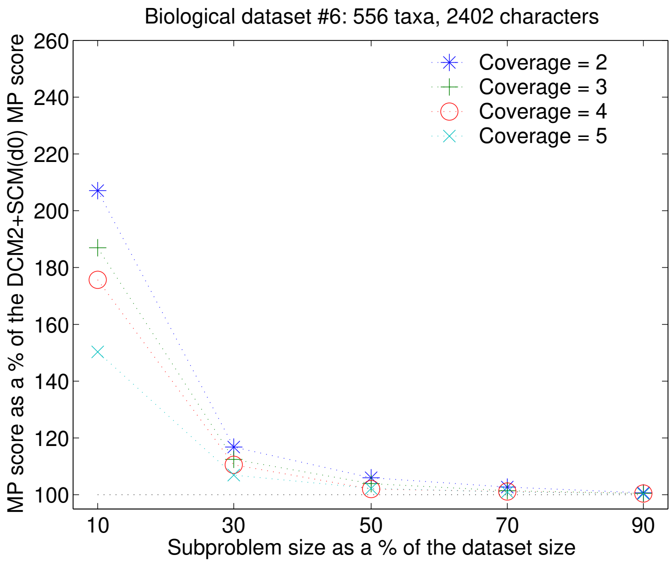

Figure 22: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #5 of 556 taxa and 2,402 sites

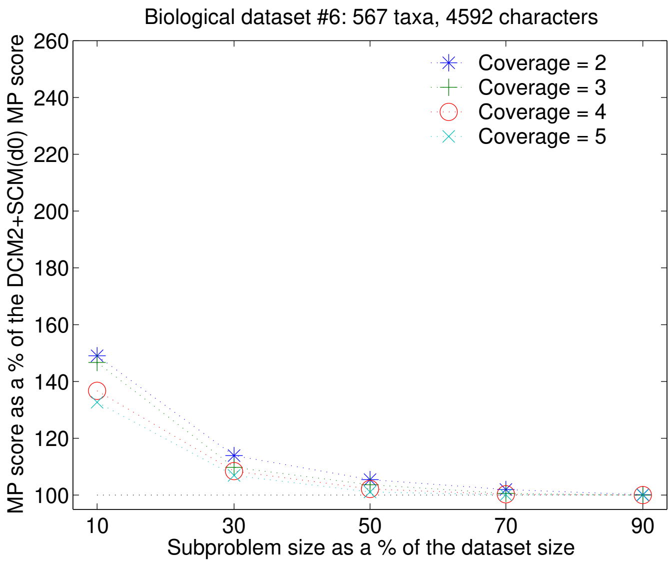

Figure 23: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #6 of 567 taxa and 4,592 sites

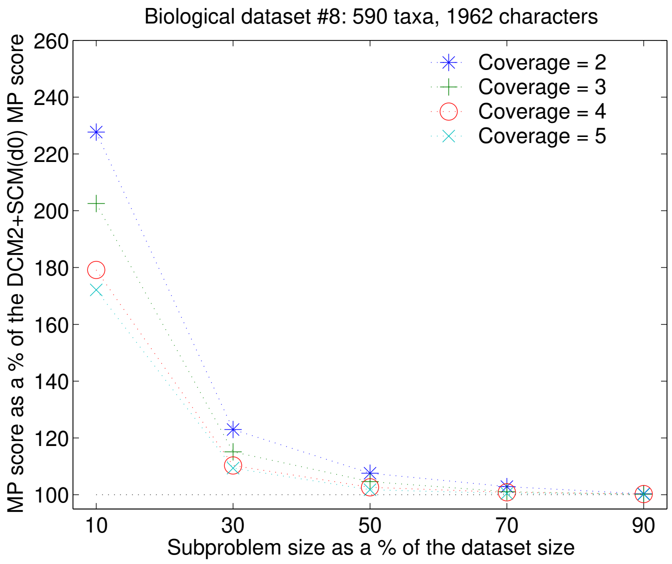

Figure 24: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #7 of 590 taxa and 1,962 sites

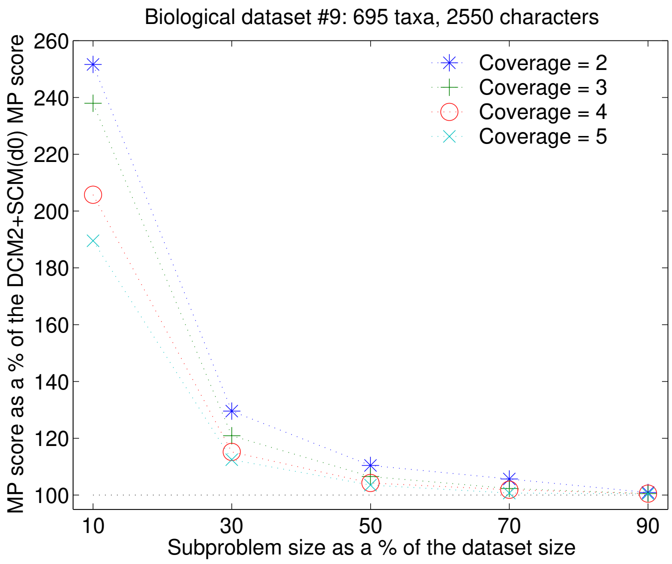

Figure 25: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #8 of 695 taxa and 2,550 sites

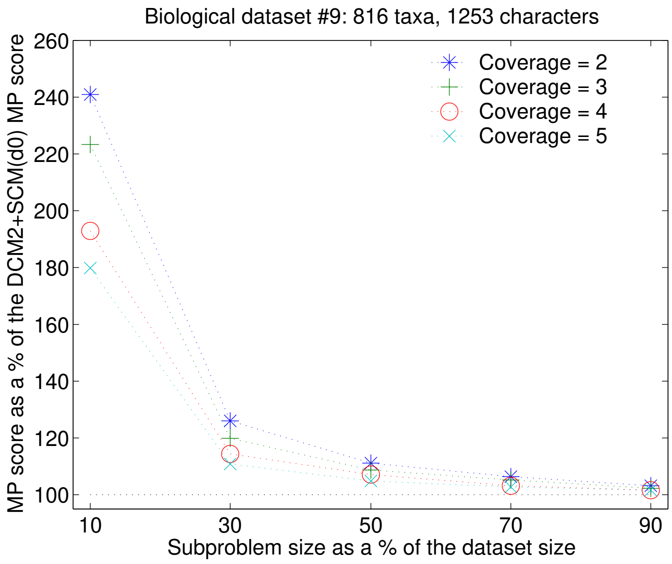

Figure 26: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #9 of 816 taxa and 1,253 sites

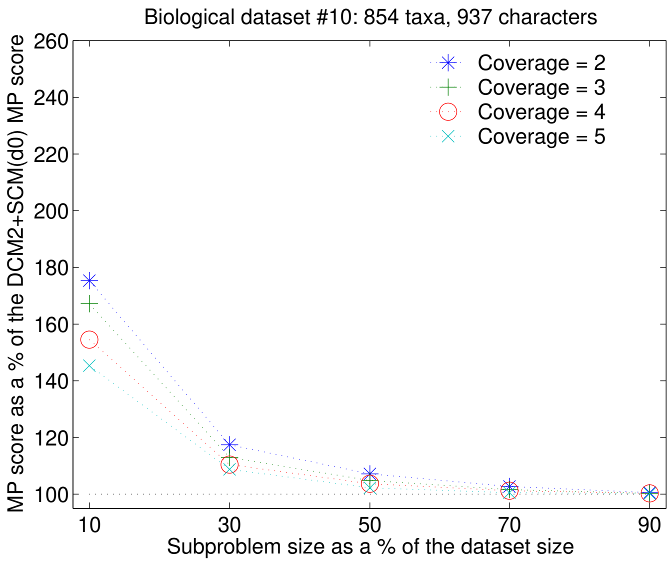

Figure 27: Comparison of RANDOM (averaged over 5 runs) with various subproblem sizes and coverages on biological dataset #10 of 854 taxa and 937 sites

Table 1: Comparison of DCM-based methods and RANDOM on biological dataset #1 of 328 taxa and 946 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 77.4% (254), max subproblem size = 87.5% (287)
#subproblems = 6, overlap size = 73.2% (240)

DCM2(d_4): Avg subproblem size = 88.2% (289), max subproblem size = 91.8% (301)
#subproblems = 3, overlap size = 82.3% (270)

DCM1(d_0): Avg subproblem size = 27.4% (90), max subproblem size = 86% (282)
#subproblems = 28

Dataset: max p-distance = 0.524

Methods	MP score	Total time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	12759	28623.62	12.47	28592.15	2	17	73.2%
DCM2+MRP(d_0)	12779	32741.62	12.47	28592.15	4124	13	87.7%
DCM2+SCM(d_4)	12749	15539.88	18.69	15500.19	1	20	77.8%
DCM2+MRP(d_4)	12764	19598.88	18.69	15500.19	4065	15	88.9%
DCM1+SCM(d_0)	12754	35010	19	34918	53	20	58.2%
DCM1+MRP(d_0)	12760	40465	19	34918	5508	20	3.1%
RANDOM(avg)	12810.6	33839.61	1.61	29558.8	4262.6	16.6	82.6%

Table 2: Comparison of DCM-based methods and RANDOM on biological dataset #2 of 439 taxa and 2,461 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 81% (355), max subproblem size = 98.6% (433)
#subproblems = 3, overlap size = 71.3% (313)

DCM2(d_4): Avg subproblem size = 98% (430), max subproblem size = 98.4% (432)
#subproblems = 2, overlap size = 96% (421)

DCM1(d_0): Avg subproblem size = 43% (188), max subproblem size = 98.6% (433)
#subproblems = 4

Dataset: max p-distance = 0.649

Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	41302	56195.55	15.29	56050.26	2	128	79.8%
DCM2+MRP(d_0)	41345	60867.55	15.29	56050.26	4711	91	91.3%
DCM2+SCM(d_4)	41294	60598.28	53.78	60428.50	1	115	83.9%
DCM2+MRP(d_4)	41297	65124.28	53.78	60428.50	4549	93	88.3%
DCM1+SCM(d_0)	41464	42435	28	42150	150	107	79.4%
DCM1+MRP(d_0)	41297	46988	28	42150	4588	222	29.1%
RANDOM(avg)	41420.2	73013.43	2.03	68082.8	4785.6	143	87.7%

Table 3: Comparison of DCM-based methods and RANDOM on biological dataset #3 of 476 taxa and 1,008 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 77.6% (370), max subproblem size = 85.1% (405)
#subproblems = 4, overlap size = 70.2% (334)

DCM2(d_4): Avg subproblem size = 91% (433), max subproblem size = 91.6% (436)
#subproblems = 2, overlap size = 81.7% (389)

DCM1(d_0): Avg subproblem size = 39.7% (189), max subproblem size = 77.3% (368)
#subproblems = 50

Dataset: max p-distance = 0.45

Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	17793	30891.94	34.98	30723.96	3	130	67.2%
DCM2+MRP(d_0)	17824	36173.94	34.98	30723.96	5315	100	82.5%
DCM2+SCM(d_4)	17784	20599.19	39.97	20430.22	1	128	71.5%
DCM2+MRP(d_4)	17797	25751.19	39.97	20430.22	5181	100	79.3%
DCM1+SCM(d_0)	17829	179992	39	179662	149	142	43.6%
DCM1+MRP(d_0)	17829	195248	39	179662	15476	71	45.7%
RANDOM(avg)	17891.2	41460.78	2.18	35169	6164	125.6	71.7%

Table 4: Comparison of DCM-based methods and RANDOM on biological dataset #4 of 500 taxa and 1,398 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 86% (430), max subproblem size = 91.2% (456)
#subproblems = 3, overlap size = 79% (395)

DCM2(d_4): Avg subproblem size = 91.7% (459), max subproblem size = 94.4% (472)
#subproblems = 3, overlap size = 87.6% (438)

DCM1(d_0): Avg subproblem size = 31.8% (159), max subproblem size = 88% (440)
#subproblems = 23

Dataset: max p-distance = 0.184

Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	16535	19999.6	36.90	19860.70	2	100	73.0%
DCM2+MRP(d_0)	16563	25301.6	36.90	19860.70	5312	92	90.1%
DCM2+SCM(d_4)	16538	28230.84	68.10	28057.74	4	101	71.8%
DCM2+MRP(d_4)	16567	33581.84	68.10	28057.74	5363	93	88.9%
DCM1+SCM(d_0)	16564	49709	54	49375	211	69	61.6%
DCM1+MRP(d_0)	16545	57312	54	49375	7831	52	14.5%
RANDOM(avg)	16576.2	26125.91	2.31	20741.4	5285.4	96.8	85.6%

Table 5: Comparison of DCM-based methods and RANDOM on biological dataset #5 of 556 taxa and 2,402 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 70.1% (390), max subproblem size = 86.7% (482)
#subproblems = 5, overlap size = 62.6% (348)

DCM2(d_4): Avg subproblem size = 94.1% (523), max subproblem size = 96.9% (539)
#subproblems = 3, overlap size = 91.2% (507)

DCM1(d_0): Avg subproblem size = 32% (177), max subproblem size = 82.4% (458)
#subproblems = 41

Dataset: max p-distance = 0.31

Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	17547	60306.79	96.89	60070.90	5	134	51.9%
DCM2+MRP(d_0)	17544	67844.79	96.89	60070.90	7556	121	66.4%
DCM2+SCM(d_4)	17525	65932.1	105.27	65628.83	3	195	53.0%
DCM2+MRP(d_4)	17557	71575.1	105.27	65628.83	5718	123	73.6%
DCM1+SCM(d_0)	17651	160336	109	159916	217	94	36.9%
DCM1+MRP(d_0)	17562	173806	109	159916	13645	136	65.3%
RANDOM(avg)	17686.2	69820.87	3.47	62132.8	7524.8	159.8	51.4%

Table 6: Comparison of DCM-based methods and RANDOM on biological dataset #6 of 567 taxa and 4,592 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 55.8% (317), max subproblem size = 98.9% (561)
#subproblems = 2, overlap size = 11.6% (66)

DCM2(d_4): Avg subproblem size = 94.2% (534), max subproblem size = 98.9% (561)
#subproblems = 2, overlap size = 88.4% (501)

DCM1(d_0): Avg subproblem size = 19.4% (110), max subproblem size = 98.9% (561)
#subproblems = 6

Dataset: max p-distance = 0.15

Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	45105	20922.03	84.38	20754.65	1	83	92.7%
DCM2+MRP(d_0)	45106	26530.03	84.38	20754.65	5613	78	93.4%
DCM2+SCM(d_4)	45107	37622.8	191.99	37310.81	1	119	83.5%
DCM2+MRP(d_4)	45134	43255.8	191.99	37310.81	5659	94	93.8%
DCM1+SCM(d_0)	45125	20668	114	20187	293	74	93.1%
DCM1+MRP(d_0)	45113	26532	114	20187	6074	157	36.3%
RANDOM(avg)	45478.2	28404.19	2.39	21347.8	6980.8	73.2	58.1%

Table 7: Comparison of DCM-based methods and RANDOM on biological dataset #7 of 590 taxa and 1,962 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 76.3% (450), max subproblem size = 93.7% (553)
#subproblems = 3, overlap size = 64.4% (380)

DCM2(d_4): Avg subproblem size = 88.2% (521), max subproblem size = 97.5% (575)
#subproblems = 3, overlap size = 82.4% (486)

DCM1(d_0): Avg subproblem size = 36.8% (217), max subproblem size = 92% (540)
#subproblems = 19

Dataset: max p-distance = 0.382

Methods	MP score	Time of each component in seconds					Tree resol. before OTR
		Total	Decomp.	Base	Merge	OTR	
DCM2+SCM(d_0)	25011	34843.18	105.13	34603.05	3	132	70.0%
DCM2+MRP(d_0)	25070	40608.18	105.13	34603.05	5818	82	82.3%
DCM2+SCM(d_4)	25010	47732.88	33.33	47546.55	3	150	62.7%
DCM2+MRP(d_4)	25065	53442.88	33.33	47546.55	5767	96	81.4%
DCM1+SCM(d_0)	25037	90095	161	89473	296	165	58.4%
DCM1+MRP(d_0)	25041	98619	161	89473	8750	235	24.5%
RANDOM(avg)	25103.6	44648.14	2.74	38274.2	6256.2	115	72.8%

Table 8: Comparison of DCM-based methods and RANDOM on biological dataset #8 of 695 taxa and 2,550 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 49.5% (344), max subproblem size = 99.6% (692)
#subproblems = 3, overlap size = 24.3% (169)

DCM2(d_4): Avg subproblem size = 96% (667), max subproblem size = 99.3% (690)
#subproblems = 5, overlap size = 95% (660)

DCM1(d_0): Avg subproblem size = 32.5% (226), max subproblem size = 99.3% (690)
#subproblems = 5

Dataset: max p-distance = 0.219

Methods	MP score	Time of each component in seconds					Tree resol. before OTR
		Total	Decomp.	Base	Merge	OTR	
DCM2+SCM(d_0)	12837	50778.14	49.52	50513.62	2	213	71.7%
DCM2+MRP(d_0)	12845	58460.14	49.52	50513.62	7729	168	78.8%
DCM2+SCM(d_4)	12856	191558.16	418.27	190843.89	8	288	40.0%
DCM2+MRP(d_4)	12860	200001.16	418.27	190843.89	8568	171	76.2%
DCM1+SCM(d_0)	12897	50628	127	49658	504	339	56.6%
DCM1+MRP(d_0)	12863	58674	127	49658	8472	417	15.6%
RANDOM(avg)	13711.4	48856.73	2.33	32753	15749.8	351.6	63.6%

Table 9: Comparison of DCM-based methods and RANDOM on biological dataset #9 of 816 taxa and 1,253 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 36% (296), max subproblem size = 60% (486)							
#subproblems = 3, overlap size = 4.4% (36)							
DCM2(d_4): Avg subproblem size = 85% (693), max subproblem size = 85% (696)							
#subproblems = 2, overlap size = 70% (571)							
DCM1(d_0): Avg subproblem size = 21.4% (175), max subproblem size = 52% (424)							
#subproblems = 10							
Dataset: max p-distance = 0.46							
Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	30138	23670.04	126.01	23277.03	1	266	68.0%
DCM2+MRP(d_0)	30188	42827.04	126.01	23277.03	18997	427	27.2%
DCM2+SCM(d_4)	30157	62627.64	185.22	62041.42	3	398	59.5%
DCM2+MRP(d_4)	30209	72896.64	185.22	62041.42	10440	230	72.8%
DCM1+SCM(d_0)	30337	41298	124	40495	308	371	53.6%
DCM1+MRP(d_0)	30213	55686	124	40495	14676	391	26.7%
RANDOM(avg)	33038	49859.5	1.50	27573	21958	327	44.0%

Table 10: Comparison of DCM-based methods and RANDOM on biological dataset #10 of 854 taxa and 937 sites. Also shown are statistics on the subproblem decompositions from DCM1 and DCM2, and the maximum p-distance of the dataset

DCM2(d_0): Avg subproblem size = 90.3% (771), max subproblem size = 93.4% (798)							
#subproblems = 3, overlap size = 85.5% (730)							
DCM2(d_4): Avg subproblem size = 84% (717), max subproblem size = 98.8% (844)							
#subproblems = 2, overlap size = 68% (580)							
DCM1(d_0): Avg subproblem size = 27.1% (232), max subproblem size = 93.4% (798)							
#subproblems = 35							
Dataset: max p-distance = 0.39							
Methods	MP score	Time of each component in seconds					Tree resol.
		Total	Decomp.	Base	Merge	OTR	before OTR
DCM2+SCM(d_0)	23029	102941.36	456.41	102017.95	6	461	66.3%
DCM2+MRP(d_0)	23057	113200.36	456.41	102017.95	10469	257	88.0%
DCM2+SCM(d_4)	23035	58571.12	235.88	58012.24	3	320	69.3%
DCM2+MRP(d_4)	23061	68801.12	235.88	58012.24	10314	239	82.5%
DCM1+SCM(d_0)	23087	238123	547	235959	986	631	54.8%
DCM1+MRP(d_0)	23059	267326	547	235959	30396	424	60.0%
RANDOM(avg)	23064	116956.9	6.70	105423.2	11169	358	83.8%