The Role of Diverse Populations in Phylogenetic Analysis

Tiffani L. Williams
Department of Computer Science
Texas A&M University
College Station, TX 77843
tlw@cs.tamu.edu

Marc L. Smith
Department of Computer Science
Colby College
Waterville, ME 04901
mlsmith@colby.edu

ABSTRACT

The most popular approaches for reconstructing phylogenetic trees attempt to solve NP-hard optimization criteria such as maximum parsimony (MP). Currently, the best-performing heuristic for reconstructing MP trees is Recursive-Iterative DCM3 (Rec-I-DCM3), which uses a single tree (or solution) to guide its way through an exponentially-sized tree space. To improve performance further, we designed Cooperative Rec-I-DCM3, a population-based approach for utilizing a population of Rec-I-DCM3 trees.

We compare the performance of Cooperative Rec-I-DCM3 to Rec-I-DCM3 on four large biological datasets. Of particular interest is Cooperative Rec-I-DCM3's selection criteria for maintaining a population of solutions. Our experiments reveal that diverse populations outperform Rec-I-DCM3 in terms of average rates of convergence to best-known MP scores. To achieve greater performance, we designed an elitist population strategy, in which each solution's tree score matches that of the best score found in each generation. The elitist strategy was by far the worst overall performer in our experiments. Hence, being greedy is not always the best approach. Instead, a population of diverse solutions allows our cooperative algorithm to achieve the greatest performance improvements.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods; J.3 [**Life and Medical Sciences**]: Biology and genetics

General Terms

Algorithms, Experimentation, Performance

Keywords

phylogeny reconstruction, maximum parsimony, memetic algorithms

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

1. INTRODUCTION

Given a collection of n organisms (or taxa), the objective of phylogeny reconstruction is to produce an evolutionary tree describing the evolutionary relationships between the organisms (or taxa). Here, the leaves of the tree denote the taxa of interest, and the edges denote the evolutionary relationships between the organisms. A phylogenetic tree is a hypothesis of the evolutionary history of the n taxa under study. Hence, evolutionary trees attempt to predict the past with information (e.g., biomolecular sequences, morphological data) from current-day species. The ultimate challenge is to reconstruct the *Tree of Life*, the evolutionary history of all-known organisms.

Unfortunately, current approaches are several orders of magnitude away from being able to infer the *Tree of Life*; estimates indicate it consists of 10 to 100 million organisms. Here, we tackle the problem by applying evolutionary computing techniques (e.g., selection and recombination) to a population of evolutionary trees, each reconstructed by a local search heuristic. Our approach could be categorized as a *memetic algorithm* [14], which combines a population-based global search and an individual local search to exploit problem-domain knowledge.

The novelty of our cooperative approach—as applied to phylogenetics—reflects our curiosity as to whether maintaining a population of tree solutions outperforms heuristics that manipulate a single tree solution. A population-based approach seems more suited to balance the orthogonal search objectives of *exploration* and *exploitation*, where exploration helps the search avoid getting trapped in local optima and exploitation investigates promising regions of the search space more effectively.

We developed Cooperative Rec-I-DCM3 [21], a memetic algorithm that maintains a population of Rec-I-DCM3 [18] trees to search the tree landscape. Rec-I-DCM3 reconstructs trees using maximum parsimony (MP) as an optimization criterion. MP is a commonly used criteria for reconstructing evolutionary trees that is based on Occam's Razor, which states that the simplest explanation that accounts for the data is the best. Hence, the tree that explains the data with the fewest evolutionary events (i.e., mutations) is the one that is preferred under MP. Experiments have shown that Rec-I-DCM3 consistently outperforms traditional approaches such as parsimony ratchet [16] and TNT [7] on large biological datasets.

Our study compares the performance of Cooperative Rec-I-DCM3 to Rec-I-DCM3 on four datasets ranging from 921 to 4,114 taxa. Our study addresses the following questions.

1. Can a cooperative approach to inferring phylogenetic trees outperform a highly-tuned local search heuristic such as Rec-I-DCM3?
2. What effect do different cooperation strategies have on performance? In particular, does a population that consists entirely of elitist solutions (i.e., the best solutions in terms of their tree score) outperform a diverse strategy, which is composed of a heterogeneous range of tree scores?

The elitist (greedy) strategy was motivated, in part, by techniques employed in PAUP* [20], a widely-used phylogenetic application. Our experimental results show that Cooperative Rec-I-DCM3 with diverse tree scores outperforms Rec-I-DCM3 in terms of average rate of convergence to the best-known scores. Moreover, our results clearly demonstrate that a diverse population outperforms its elitist counterpart. In many cases, the elitist strategy performs worse than Rec-I-DCM3. Hence, being greedy is not always the best approach. By maintaining a population of diverse solutions, our cooperative algorithm is able to more fully reach its potential.

Related Work. Evolutionary computing is not a stranger to phylogenetic algorithms. While we are unaware of other phylogenetic techniques that use a memetic-based approach, there are several phylogenetic heuristics that employ genetic algorithms [1, 3, 11, 12, 17]. Unfortunately, the performance of these genetic algorithms are often not compared to the top performing phylogenetic heuristics. Thus, it is difficult to evaluate the performance of the algorithms when the top competitors are missing. We explicitly compare our cooperative algorithm to Rec-I-DCM3, which is the best MP algorithm available. Moreover, Rec-I-DCM3 has been extensively tested against other MP approaches such as TNT and parsimony ratchet and has consistently outperformed them on large datasets. Besides MP, maximum likelihood (ML) is another highly regarded optimization criterion; but its feasibility is limited to much smaller sets of taxa than the datasets analyzed in our experiments.

2. BACKGROUND

2.1 Maximum parsimony

The MP problem seeks the tree T with the minimum length. In biological terms this is the same as seeking the tree with the smallest number of point mutations for the data. MP is an NP-hard problem [5], but the problem of scoring a fixed tree is polynomial [4]. MP heuristics use polynomial time algorithms for scoring individual trees along with techniques for moving through the exponentially-sized tree space.

Formally, given two sequences a and b of the same length, the *Hamming distance* between them is defined as $|\{i : a_i \neq b_i\}|$ and denoted as $H(a, b)$. Let T be a tree whose nodes are labeled by sequences of length k over Σ, and let $H(e)$ denote the Hamming distance of the sequences at each endpoint of edge e. Let $E(T)$ be set of edges in T. The *parsimony score* of the tree T is $\sum_{e \in E(T)} H(e)$.

2.2 Tree-Bisection and Reconnection (TBR)

Iterative improvement methods are some of the most popular heuristics in phylogeny reconstruction. A fast technique is used to find an initial tree, then a local search mechanism is applied repeatedly to find trees with a better score. One popular local search technique is based on using *Tree-Bisection and Reconnection (TBR)* [13] moves to locate better scoring trees. In a TBR operation, an edge is removed from the given tree T and each pair of edges touching each endpoint merged, thereby creating two subtrees, t and $T - t$; the two subtrees are then reconnected by subdividing two edges (one in each subtree) and adding an edge between the newly introduced nodes.

2.3 Disk-Covering Methods (DCMs)

Disk-Covering Methods (DCMs) [8, 9, 15, 18] use a divide-and-conquer technique for reconstructing phylogenetic trees. DCMs consist of four main stages. First, divide the dataset into overlapping subsets (or subproblems), which are sets of taxa with nonempty intersections. Secondly, solve the subproblems. Here, any phylogenetic method of choice can be applied to solve the subproblems from the first stage. Each such solution is a tree whose leaf nodes correspond to one of the sets of taxa from stage one. Third, merge the solutions back together, recombining each of the solved subtrees back into a single tree. The hope is, the resulting tree at this stage is in someway better than the tree we began with during the first stage. Chances are, however, the resulting tree is also multifurcating. This presents a problem, because many existing phylogeny reconstruction techniques require bifurcating trees. Therefore, the fourth stage of a DCM consists of refining the tree from stage three so that it is a bifurcation.

While all DCMs consist of the above steps, variations result from different strategies during the first stage. We can strategize how to decompose subsets of taxa based on existing inter-taxa relationship measures, such as pairwise distances; or even leverage information from subtrees of the given tree. Regardless of the method of decomposition (which has a significant effect on overall performance of the DCM), the remaining three stages all work the same way.

2.4 Consensus Methods

A maximum parsimony analysis typically produces a collection of trees with the same score. In these circumstances, the standard practice is to output a single tree (or consensus tree) that summarizes the result of the search. Bryant provides an excellent overview of the wealth of consensus techniques available to summarize the results of a phylogenetic analysis [2]. However, we have another purpose for considering consensus techniques. Here, we use consensus techniques as a mechanism for creating new trees to explore during a phylogenetic analysis. It forms the major component of the recombination phase of our cooperative algorithm (see Section 3).

The two most popular techniques, strict and majority trees, use the notion of splits to construct the resulting consensus tree. Since a phylogenetic tree is an unrooted tree, every leaf is identified with a unique taxon (or sequence). Removing a branch (edge) of an unrooted tree divides the tree into two connected parts. If A is the group of taxa on one side of the branch and B is the group of taxa on the other, then $A|B$ is said to be the split corresponding to that branch. Given a collection of unrooted trees, the *strict consensus tree* contains exactly those splits common to all the trees in the collection (see Figure 1). The *majority rule tree*

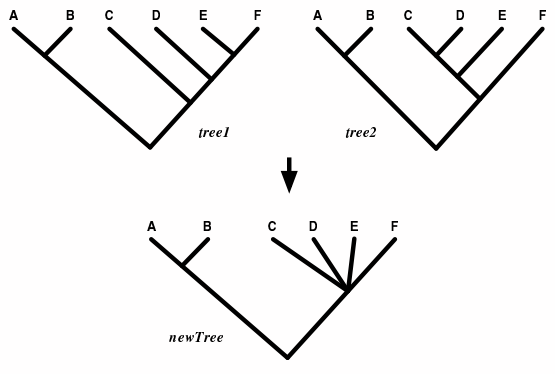


Figure 1: Strict-consensus tree. A strict consensus preserves only those subtrees in common between the input trees.

contains exactly those splits that appear in more than half of the input trees.

3. Cooperative Rec-I-DCM3

Cooperative Rec-I-DCM3 [21] is a new approach for reconstructing phylogenetic trees that maintains a population of cooperating solutions to guide its search for better MP trees. It consists of the following steps (see Figure 2).

1. Create a population of μ starting trees, which represents the initial population of solutions.
2. Run Rec-I-DCM3 on each of the μ trees.
3. Create a new population of μ trees by performing selection and recombination on the trees returned from the Rec-I-DCM3 search.
4. Repeat steps 2 and 3 for the required number of generations.

We describe each of these steps in more detail below.

3.1 Algorithm Details

Starting trees. Any approach may be used to populate the population with μ initial trees. In our implementation, we used an approach known as Greedy-MP (or random sequence addition) to create the μ trees. To construct a Greedy-MP tree, we first randomize the ordering of the sequences in the dataset. Afterwards, the first three taxa are used to create an unrooted binary tree, T. The fourth taxon is added to the internal edge of T that results in the best MP score. This process continues until all taxa have been added to the tree.

Local search. Although our cooperative algorithm is capable of handling any phylogenetic search technique in step 2 of the algorithm, we choose to use Rec-I-DCM3 since it is the best performing MP heuristic and thus the hardest to improve. Rec-I-DCM3 combines both recursion and iteration to provide a powerful technique for searching tree space. The recursive application of the decomposition step produces smaller and smaller subproblems until every subproblem is small enough to be solved directly. Once the dataset decomposition step is complete, subtrees are constructed for each subset and combined using the Strict Consensus Merger [8, 9] to produce a tree on the original dataset.

Selection and recombination. The selection process decides which solutions will enter the population of the next generation. The μ trees from step 2 are ranked based on their MP scores, with the best scoring MP tree having the best rank. Next, the trees are placed into sets A, B, and C based on their rank. The algorithm also keeps a list of trees with the best-scoring solution found by the current generation of the search. These elite trees are placed into set A. The top-ranking trees from step 2 are placed into set B, and the lower-ranking trees are put into set C. These sets of trees comprise the new population. However, trees in set C may be recombined with trees in $A \cup B$ to create new (and more diverse) solutions. If $t \in C$ is chosen for recombination, it will be replaced by the resulting recombined tree.

For each tree $t \in C$, there is a $p\%$ chance that it will undergo recombination with a random tree $t' \in A \cup B$. t and t' are recombined by computing their strict consensus tree, which contains all of the splits that are common between the trees. Since the strict consensus tree typically results in a multifurcating tree, it is refined into a binary tree and subjected to a global search using TBR moves.

3.2 Diverse and elitist populations

Recall that at generation i, selection places solutions in the population into one of three sets, A, which represents the best solutions found by generation i; B, which contain the top-ranking trees from generation i, and C, the lower-ranked trees from generation i. Our *diverse strategy*, sets $|A| = 1$, $|B| = 0.4\mu$, and $|C| = \mu - |A \cup B|$. Moreover, the chance of a solution in set C recombining with the high-scoring individuals in $A \cup B$ is 20%. In the *elitist strategy*, $|A| = \mu$ and $|B \cup C| = 0$. Moreover, there is no recombination in this strategy.

3.3 Cooperation and DCMs

The familiar reader may perceive apparent similarities between DCM approaches (such as Rec-I-DCM3) and our cooperative approach. Both algorithms are examples of divide-and-conquer strategies designed to boost the performance of existing algorithms. In fact, one could characterize Cooperative Rec-I-DCM3's use of a population of trees as a natural extension of subproblems in DCMs.

However, our cooperative algorithm extends the notion of decomposition and merging that is inherent in DCMs. Cooperative Rec-I-DCM3's use of a population of trees may be viewed as a partial decomposition of the exponentially large space of tree solutions. Each of the individuals in the population represents a decomposition of that space. DCMs decompose a single guide tree (or the original dataset of n sequences) into a population of overlapping subsets of the original problem. Hence, in DCMs, the population represents partial solutions to the original problem whereas in Cooperative Rec-I-DCM3 the population contains complete solutions. Although both approaches use techniques to merge solutions in the population, the purpose of merging is quite different. Cooperative Rec-I-DCM3 uses merging (or recombination) to create new, more diverse solutions. DCMs, on the other hand, require merging in order to obtain a sin-

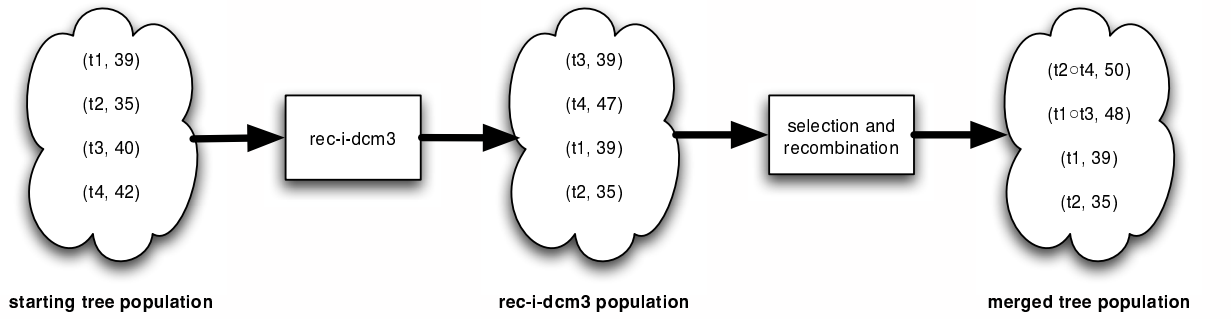


Figure 2: A generation of the Cooperative Rec-I-DCM3 algorithm. The population size (or μ) is four. Each solution (e.g., (t1, 39)) is identified by a tree and its score. Rec-I-DCM3 is applied to each tree in the starting population. The resulting trees are then subjected to selection and recombination. We represent the recombination of two trees by the composition operator ($\circ$).

gle, complete solution on the entire dataset from the partial solutions.

4. EXPERIMENTAL METHODOLOGY

Datasets. Our experiments compared the performance of the algorithms on four biological datasets ranging from 921 to 4,114 sequences. Below, we provide the details of each dataset along with their best-known score under maximum parsimony since the optimal score is not known. Each of these datasets were obtained from the recent analysis of Rec-I-DCM3 by Roshan et al. [18].

1. A set of 921 aligned Avian Cytochrome b DNA sequences (713 sites) [10]. A best score of 40,494 was established by Cooperative Rec-I-DCM3 on this dataset.
2. A set of 1,127 aligned large subunit ribosomal RNA sequences (1078 sites) obtained from the Ribosomal rRNA database [22]. The best score of 52,056 was established by Rec-I-DCM3 and Cooperative Rec-I-DCM3 on this dataset.
3. A set of 2,000 aligned Eukaryotic sRNA sequences (1251 sites) obtained from the Gutell Lab at the Institute for Cellular and Molecular Biology, The University of Texas at Austin. Our runs of both Rec-I-DCM3 and Cooperative Rec-I-DCM3 established a best score of 74,534.
4. A set of 4,114 Actinobacteria 16s rRNA sequences (1263 sites) from the Ribosomal Database Project. The best score for this dataset is 60,887, which was established by Cooperative Rec-I-DCM3.

Experiments. All experiments consisted of five runs of Rec-I-DCM3 and Cooperative Rec-I-DCM3. We ran Rec-I-DCM3 with the recommended default settings. Hence, the maximum subproblem sizes were set to 50% of the original problem size on Dataset #1 and 25% on Datasets #2, #3, and #4. In the Cooperative Rec-I-DCM3 algorithm, Rec-I-DCM3 was executed for one iteration. Both Rec-I-DCM3 and Cooperative Rec-I-DCM3 were given sufficient time to find the best-known score. Hence, Rec-I-DCM3 ran for 1,000 iterations, and its cooperative counterpart ran for 100 generations with population sizes of 2, 4, 8, and 16 individuals.

Implementation. We used TCP Linda [19], an implementation of Gelernter's Linda [6] model of concurrency, to implement our cooperative algorithm. Our TCP Linda programs were written in the C-Linda language, which augments the C language with four primitive operations that permit process creation and access to tuple space — an associative, distributed shared memory. Rec-I-DCM3 is open-source software provided by Usman Roshan. TNT was used as the base method for Rec-I-DCM3, and we used TNT's implementation of TBR for the global search phase of our recombination operator. We used PAUP*'s implementation of strict consensus.

Platforms. Our experiments were performed on two high-performance computing clusters: an Apple Workgroup Cluster for Bioinformatics and a Linux Beowulf cluster. Both clusters are similarly configured, each consisting of four, 64-bit, dual-processor nodes (eight total CPUs) with gigabit-switched interconnects. However, the underlying hardware of the clusters is quite different. The Apple Workgroup Cluster consists of Xserve G5 nodes, each of which contains two, 2 GHz PowerPC G5 processors. Each processor contains 512 KB of L2 cache and a 1 GHz front-side bus; the two processors on each node share 4 GB of DDR 400 MHz SDRAM (16 GB total RAM across the cluster). The Linux Beowulf cluster consists of four nodes; each node contains two, 2 GHz Intel Xeon processors. Each processor contains 512 KB of L2 cache, but only a 400 MHz front-side bus; the two processors on each node share 2 GB of DDR 266 MHz SDRAM (8 GB total RAM across the cluster).

5. EXPERIMENTAL RESULTS

We compare the performance of our cooperation strategies (diverse and elitist) to Rec-I-DCM3. Typically, heuristics are evaluated by how fast good solutions can be obtained and by how far such solutions are from optimal. However, the optimal solution is unknown on the biological datasets used in this study. As a result, we plot performance in

Table 1: The average time per generation (in seconds) of Rec-I-DCM3 ($\mu = 1$) and Cooperative Rec-I-DCM3 (μ = 2, 4, 6, and 8) using a diverse population on the four biological datasets.

μ	Dataset 1 921 taxa	Dataset 2 1,127 taxa	Dataset 3 2,000 taxa	Dataset 4 4,114 taxa
1	52.66	56.20	118.41	191.37
2	55.20	59.73	161.49	254.07
4	59.33	64.22	185.43	282.33
8	115.50	103.70	337.95	468.44
16	272.92	245.61	535.79	887.45

terms of the number of steps, s, a solution is from the best known score, b, found by any MP phylogenetic analysis. More specifically, if b_i is the best score found by generation i, then $s = b_i$ - b. Best-known scores for our datasets are provided in Section 4. Moreover, all data points in the plots are the average of five runs.

5.1 Diverse populations

First, we compare the performance of Cooperative Rec-I-DCM3 using a diverse population with that of Rec-I-DCM3. Figure 3 shows the results. The plots clearly demonstrate that Cooperative Rec-I-DCM3 outperforms Rec-I-DCM3 within 100 generations. Even after 1,000 generations, Rec-I-DCM3 is unable to match Cooperative Rec-I-DCM3's performance. Both algorithms converged to the best-known score on Dataset # 2 (1,127 taxa), but Cooperative Rec-I-DCM3 also converged to the best-known score on Dataset # 3 (2,000 taxa). Larger population sizes result in increased performance, where populations sizes of 8 and 16 perform the best.

Traditional evolutionary approaches use larger population sizes than we employ in our experimental study. For our memetic-based approach, the local search phase accounts for the majority of the running time within a generation. The runtime requirement for the Rec-I-DCM3 search coupled with the large dataset sizes limit the number of individuals that can be effectively handled on our experimental platform. Table 1 shows the time required per generation for each of the algorithms under study. On the largest dataset, 1,000 generations of the Rec-I-DCM3 algorithm ($\mu = 1$) requires 53.16 hours whereas 100 generations of Cooperative Rec-I-DCM3 require 24.65 hours when $\mu = 16$.

Figure 4 compares the performance of the algorithms within a 24 hour time period on the two largest datasets. (Space limitations prevent us from showing such graphs for our smallest datasets.) The plots show that the wall-clock performance of Cooperative Rec-I-DCM3 is better than that of Rec-I-DCM3 after 24 hours. More specifically, Figure 4 shows that Rec-I-DCM3 cannot reach scores obtained by Cooperative Rec-I-DCM3 on these datasets within the allotted time.

Implementation. Table 1 shows that there is a substantial runtime penalty for using populations of size 8 and 16. This is a result of our implementation of the Cooperative Rec-I-DCM3 algorithm. Our implementation takes advantage of a parallel platform, where each solution in the population is assigned to a separate processor. Since our parallel platforms have a maximum of eight processors, on populations of size 16, the processors perform twice more work each generation, which translates into larger wall-clock times.

Moreover, there are other processes (besides the μ Rec-I-DCM3 workers) needed to manage the cooperative algorithm. For example, we use a startup worker, who oversees the entire computation, and a merger worker, who is responsible for selection and recombination. Each of these workers is assigned to a separate processor as well. So, even for $\mu = 8$, our current implementation requires some processors to perform more work than others—especially during the local search phase of the algorithm. Our implementation can be improved by more efficient use of idle workers. In general, if μ is greater than the number of processors, p, each processor will on average have to process $\frac{\mu}{p}$ trees. The results shown in Figure 3 assumes that $\mu = p$ and the overhead associated with creating a new population is negligible.

Even with the above implementation and resource concerns, Cooperative Rec-I-DCM3 still outperforms Rec-I-DCM3. Better wall-clock performance can be expected with a more efficient implementation and larger parallel platforms.

5.2 Elitist populations

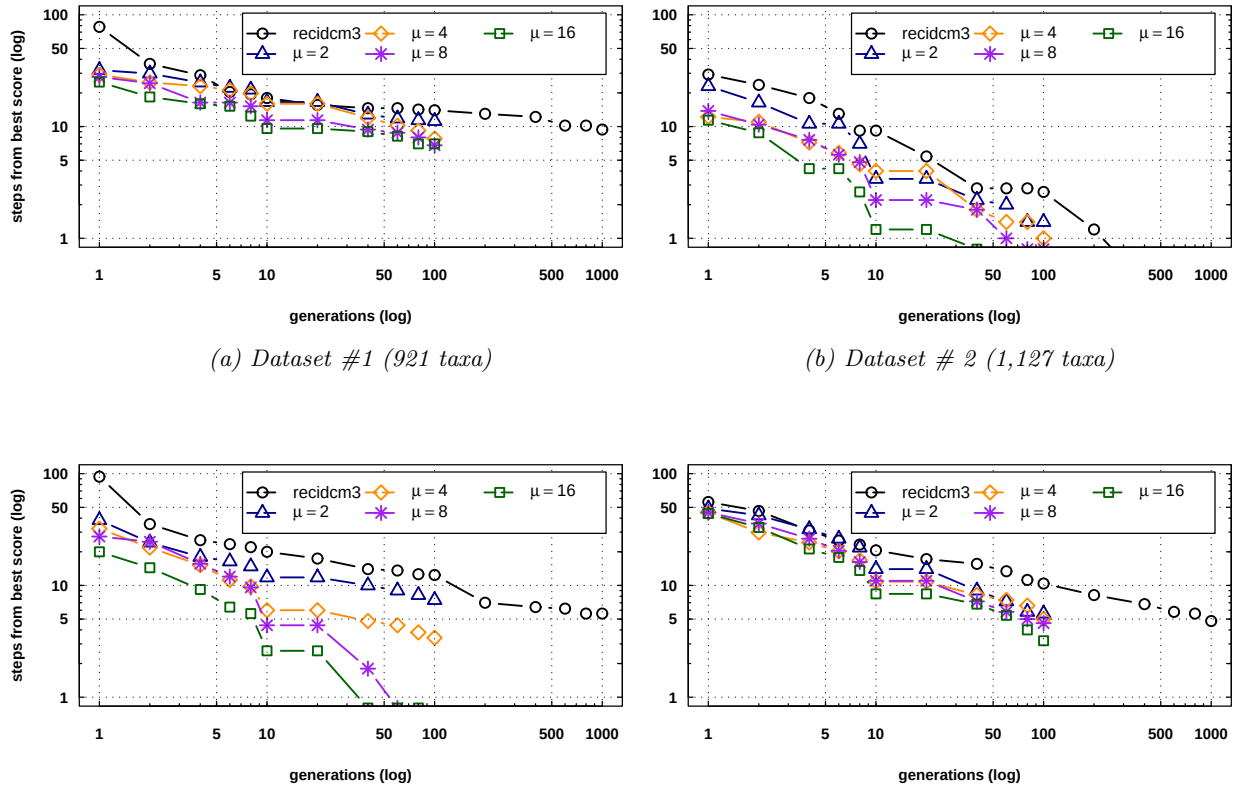
Next, we explored the performance of Cooperative Rec-I-DCM3 when it maintains a population of the best (elite) solutions found during the search. Figure 5 compares the performance of an elitist strategy with that of Rec-I-DCM3. The plot shows that such an elitist approach can be detrimental to the overall performance of the algorithm. The elitist strategy when used with small populations sizes ($\mu = 1$ and 2) is especially bad; this strategy performs worse than Rec-I-DCM3. On datasets where Cooperative Rec-I-DCM3 with a diverse group of solutions had no trouble converging to the best known score with $\mu = 8$ and 16 (i.e., see Figures 3(b) and 3(c)), the elitist strategy performed poorly. In fact, it's performance was worse than that of Rec-I-DCM3.

This result has other implications, especially in the context of traditional phylogenetic searches. Often, these searches maintain a list of the best scoring trees found. When a better tree is found, the current list of best trees is replaced with the newly found best scoring tree. This is exactly the elitist strategy that we employ in our experiments. Our results suggest that such a strategy should be revisited as it may be limiting the performance of a phylogenetic analysis.

5.3 Diversity and best trees

For the diverse strategy, we look at how often the best scoring tree by generation i (b_i) produces another best scoring tree (b_{i+1}) in the next generation. In other words, we plot the percentage of time that $b_{i+1} \leq b_i$. For the elitist strategy, this percentage is always 100% since the only trees in the population are the best trees found so far. In our diverse strategy, b_i always appears in the population at least once. b_i appears multiple times if a new best score is found (i.e., $b_i < b_{i-1}$) since it is both a new best tree and a top-ranking tree (see Section 3.1).

Figure 6 provides some insight as to why the diverse strategy outperforms its elitist counterpart. For the most part, b_i finds other best trees 40%–70% of the time. On Dataset #2 when $\mu = 16$, b_i is responsible for producing better or equal scores about 10% of the time. Hence, the other lower-scoring trees are responsible for 90% of the search improvement in terms of best trees found! Our results demonstrate that access to the best tree is an important component of an effective search. Lesser solutions when coupled with the best,



(a) Dataset #1 (921 taxa)

(b) Dataset # 2 (1,127 taxa)

(c) Dataset #3 (2,000 taxa)

(d) Dataset #4 (4,114 taxa)

Figure 3: Performance of Rec-I-DCM3 and Cooperative Rec-I-DCM3 using a diverse population strategy algorithms on the four datasets under study. μ represents the population size used by the Cooperative Rec-I-DCM3 algorithm. Each data point is the average of five runs.

are responsible for a significant proportion (at least 30%) of the performance improvement exhibited by the diverse strategy.

6. CONCLUSIONS AND FUTURE WORK

Our study investigates the role that diversity plays in the performance of a phylogenetic analysis. Specifically, we compare Cooperative Rec-I-DCM3, a population-based approach that uses a cooperating pool of solutions to guide the search, and Rec-I-DCM3, the best performing MP heuristic. We consider two population retention strategies—a *diverse strategy* that maintains a heterogeneous mix of tree scores and an *elitist strategy*, which only keeps the best scoring trees found during a search.

Our experimental results show that Cooperative Rec-I-DCM3 using a diverse selection strategy consistently outperforms Rec-I-DCM3. However, our results also reveal that the elitist strategy is the worst overall performer. Hence, a cooperative approach that retains elite solution leads to poor performance. This suggests that traditional phylogenetic searches that employ an elitist approach to maintaining a list of trees to explore should be revisited. In both strategies, larger population sizes translate into increased performance.

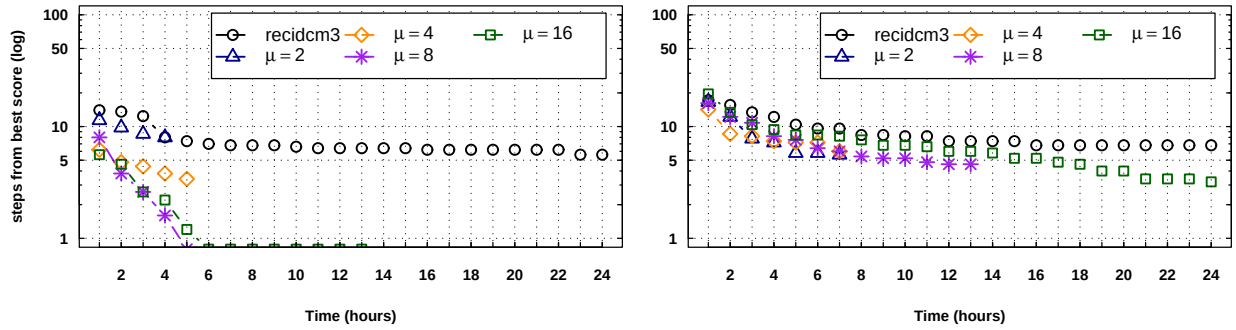
Of course, there is much future work still to be done. Since population size is an important factor in improving performance, population sizes larger than 16 should be investigated. Inefficiencies in Cooperative Rec-I-DCM3's implementation will also be improved. Other future plans include using other heuristics in our cooperative framework besides Rec-I-DCM3. Of particular interest are heuristics for maximum likelihood. Lastly, we are exploring other measures to analyze the performance of a phylogenetic analysis.

7. ACKNOWLEDGMENTS

This work was started while Williams was at the Radcliffe Institute of Advanced Study, and Smith was on sabbatical leave from Colby College. Usman Roshan provided the code for Rec-I-DCM3 and the datasets used in this study. Lastly, we thank the anonymous reviewers for their very helpful comments.

8. REFERENCES

[1] M. J. Brauer, M. T. Holder, L. A. Pries, D. J. Zwickl, P. O. Lewis, and D. M. Hillis. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Mol. Biol. Evol.*, 19(10):1717–1726, 2002.



(a) Dataset #3 (2,000 taxa) *(b) Dataset #4 (4,114 taxa)*

Figure 4: Wall-clock performance of Rec-I-DCM3 and Cooperative Rec-I-DCM3 using a diverse population strategy on the two largest datasets. μ represents the population size used by the Cooperative Rec-I-DCM3 algorithm. Since Rec-I-DCM3 was run for 1,000 generations and Cooperative Rec-I-DCM3 ran for 100 generations, some analyses were able to complete before the 24 hour time period. Each data point is the average of five runs.

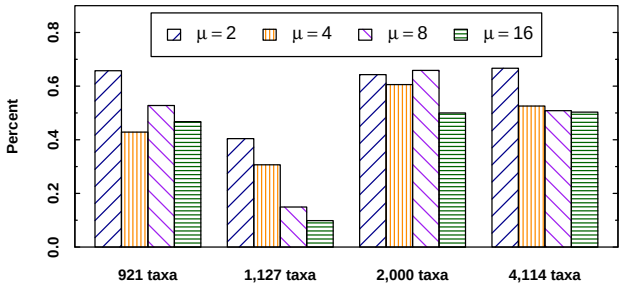


Figure 6: Percentage of time that a best scoring tree produces a solution with the same or better score in a population composed of diverse individuals.

[2] D. Bryant. A classification of consensus methods for phylogenetics. In M. Janowitz, F. Lapointe, F. McMorris, B. Mirkin, and F. Roberts, editors, *Bioconsensus*, volume 61 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 163–184. American Mathematical Society, DIMACS, 2003.

[3] C. B. Congdon. Gaphyl: An evolutionary algorithms approach for the study of natural evolution. In *Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, NY, July 2002.

[4] W. M. Fitch. Toward defining the course of evolution: minimal change for a specific tree topology. *Syst. Zool.*, 20:406–416, 1971.

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

[6] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1), Jan. 1985.

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

[8] D. Huson, S. Nettles, and T. Warnow. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *Journal of Computational Biology*, 6:369–386, 1999.

[9] D. Huson, L. Vawter, and T. Warnow. Solving large scale phylogenetic problems using DCM2. In *Proc. 7th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 118–129. AAAI Press, 1999.

[10] K. P. Johnson. Taxon sampling and the phylogenetic position of passeriformes: Evidence from 916 avian cytochrome b sequences. *Systematic Biology*, 50(1):128–136, 2001.

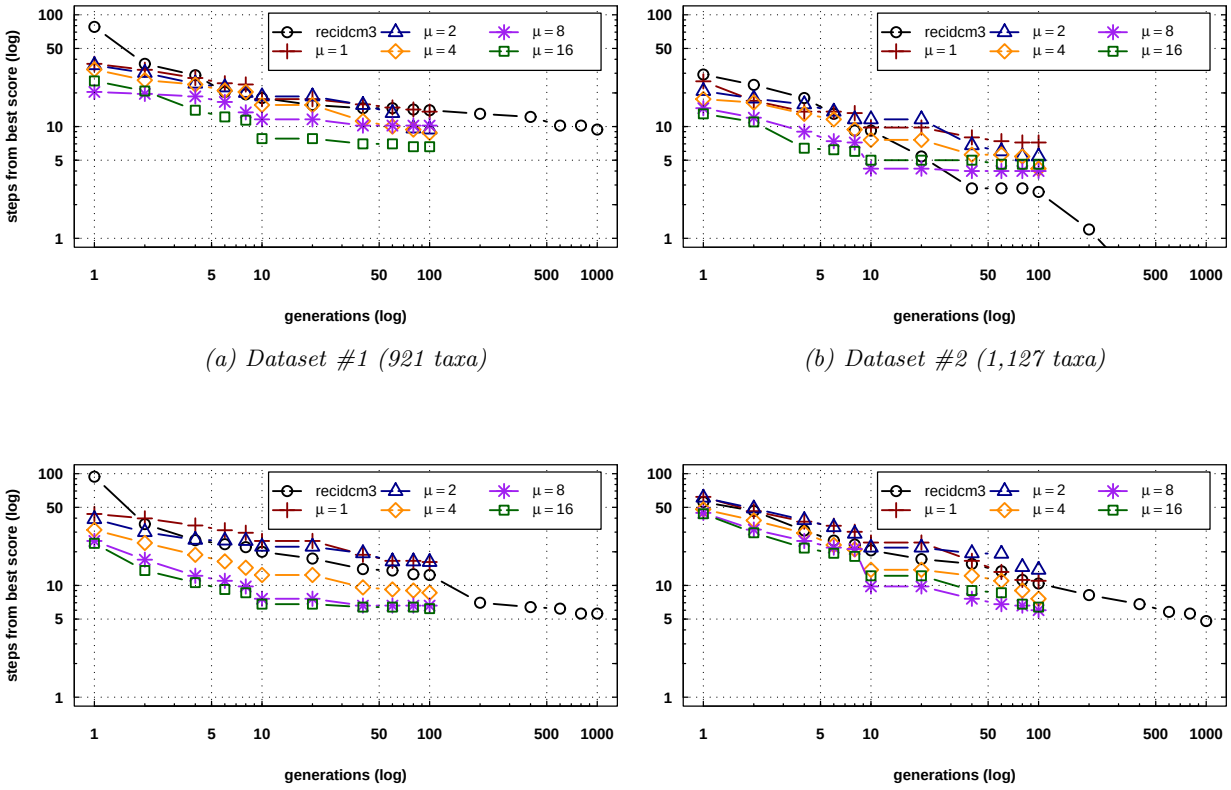
[11] A. R. Lemmon and M. C. Milinkovitch. The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. *PNAS*, 99(16):10516–10521, 2002.

[12] P. O. Lewis. A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. *Mol. Biol. Evol.*, 15(3):277–283, 1998.

[13] D. Maddison. The discovery and importance of multiple island of most parsimonous trees. *Syst. Bio.*, 42(2):200–210, 1991.

[14] P. A. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memtic algorithms. Technical report, Caltech, 1989.

[15] 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 (ISMB'01)*, volume 17 of *Bioinformatics*, pages S190–S198. Oxford Univeristy Press, 2001.



(a) Dataset #1 (921 taxa)

(b) Dataset #2 (1,127 taxa)

(c) Dataset #3 (2,000 taxa)

(d) Dataset #4 (4,114 taxa)

Figure 5: Performance of Rec-I-DCM3 and the Cooperative Rec-I-DCM3 using an elitist population strategy. μ represents the population size used by the cooperative algorithm. Each data point is the average of five runs.

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

[17] L. Poladian. A GA for maximum likelihood phylogenetic inference using neighbour joining as a genotype to phenotype mapping. In H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2005*, pages 415–422, Washington DC, USA, 2005. ACM Press.

[18] U. Roshan, B. M. E. Moret, T. L. Williams, and T. Warnow. Rec-I-DCM3: a fast algorithmic techniques for reconstructing large phylogenetic trees. In *Proc. IEEE Computer Society Bioinformatics Conference (CSB 2004)*, pages 98–109. IEEE Press, 2004.

[19] Scientific Computing Associates, Inc. TCP Linda. Internet Website, last accessed, July 2005. SCAI's TCP Linda URL: http://www.lindaspaces.com/products/linda.html.

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

[21] T. L. Williams and M. L. Smith. Cooperative-Rec-I-DCM3: A population-based approach for reconstructing phylogenies. In *Proc. Third IEEE Symp. on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'05)*, pages 127–134, 2005.

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