Phylogenetic Analysis of Large Sequence Data Sets

Hyrum Carroll*, Mark Clement*, Quinn Snell* and Keith Crandall†

* Computer Science Department,
Brigham Young University, Provo, Utah 84602
Email: {hdc,clement,snell}@cs.byu.edu

† Integrative Biology Department,
Brigham Young University, Provo, Utah 84602
Email: Keith_Crandall@byu.edu

***Abstract*— Phylogenetic analysis is an integral part of biological research. As the number of sequenced genomes increases, available data sets are growing in number and size. Several algorithms have been proposed to handle these larger data sets. A family of algorithms known as *disc covering methods* (DCMs), have been selected by the NSF funded CIPRes project to boost the performance of existing phylogenetic algorithms. *Recursive Iterative Disc Covering Method 3* (Rec-I-DCM3), recursively decomposes the guide tree into subtrees, executing a phylogenetic search on the subtree and merging the subtrees, for a set number of iterations. This paper presents a detailed analysis of this algorithm.**

I. Introduction

Phylogenetic analysis has become an integral part of biological research. This includes such diverse areas as human epidemiology [1], [2], viral transmission [3], biogeography [4], and systematics [5]. With the advent of high speed sequencing equipment, an increasingly large volume of sequence data is becoming available. As the volume of this data increases, scientists are able to ask important questions that were not possible with smaller data sets. For example, when a new virus is detected, it is possible to build a phylogenetic tree (phylogeny) containing all related viruses and the unknown variety in order to answer questions such as:

Where did this virus come from?

When did this virus arrive in the human population?

What are the related species from which we might derive ideas about appropriate antibodies for testing and remedies for treatment?

Has this virus been genetically modified through natural or human induced recombinant technology?

How is this virus evolving and what genetic changes occurred to allow it to successfully enter the human population?

In the case of the SARS epidemic, and others like it, treatment information must be available in days or at most weeks in order for appropriate action to be taken. To perform the necessary analysis in such cases, efficient computational algorithms are necessary to deliver accurate answers within a reasonable amount of time.

Phylogenies are trees with genetic sequences as leaf nodes, and interior nodes representing hypothetical ancestral sequences. Trees are typically scored by either summing the differences between two nodes (*Maximum Parsimony*) or by inferring the likelihood of the evolutionary changes (*Maximum Likelihood*). When large data sets are examined (greater than 10,000 sequences), the number of trees in the search space is so large that a complete search is infeasible. For example, for just 10 sequences, the number of possible trees is 2,027,025. With 1,000 sequences, the number of possible trees increases to 2.7×10^{2900}. Due to the exponential increase in possible trees, search techniques used on small data sets are not effective for data sets with thousands of sequences.

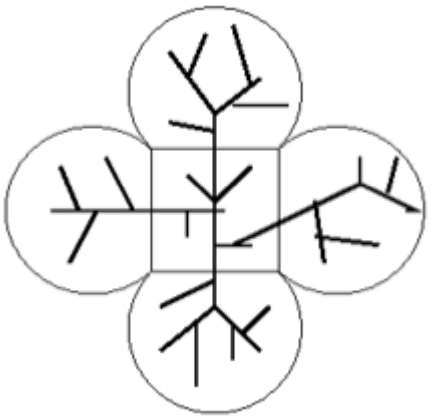

Fig. 1. Example tree decomposition performed in DCM2 and DCM3. (Graphic courtesy of Tandy Warnow)

The purpose of this paper is to analyze the performance of a particular algorithm, *Recursive Iterative Disc Covering Method 3* (Rec-I-DCM3) which has been shown to be effective at improving tree scores for large data sets.

A. Disc Covering Methods

To address the needs presented by large data sets, a family of algorithms, known as *disc covering methods* (DCM), was developed [6]–[10]. DCMs boost the performance of underlying phylogenetic algorithms by decomposing trees into subtrees (see Figure 1). Instead of executing a search on the entire tree, DCMs partition the tree into many subtrees that can be easily searched. Since the sum of the number of

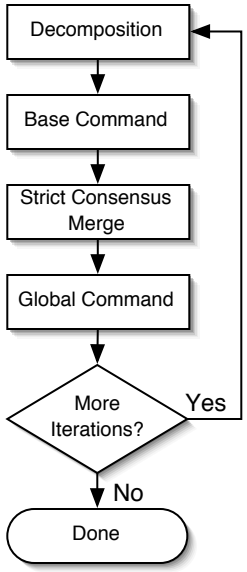

Fig. 2. Rec-I-DCM3 execution flow.

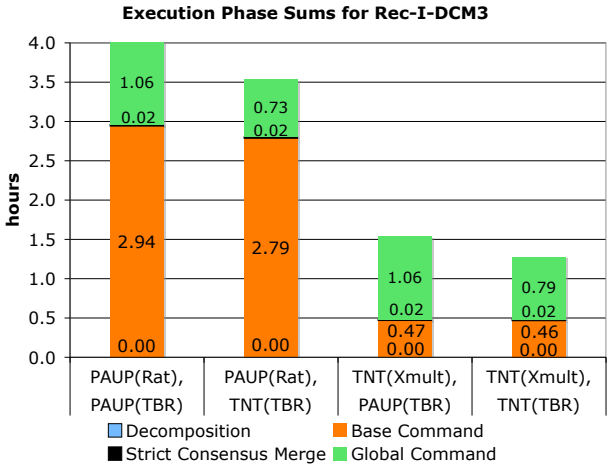

Fig. 3. Sums of Rec-I-DCM3 execution phases for one iteration. X-axis is base command, global command. (Note: the Decomposition phase executed for less than a second)

possible tree permutations for each subtree is exponentially smaller than the number of permutations of the full tree, the decomposition has the effect of exponentially reducing the search space. Searches that are intractable on the entire tree can be executed on all the subtrees in a reasonable amount of time.

The first DCM, DCM1 [6], [7], was designed to improve the performance of distance-based methods (e.g., Neighbor Joining). When Maximum Parsimony (MP) searches are used, DCM1 produces too many subtrees. DCM2 was developed in response to this and produces larger subsets based on a distance matrix for MP analysis. *Recursive Iterative DCM3* (Rec-I-DCM3) [9] improved on previous methods and was selected by the NSF funded CIPRes project [11] as a major algorithm for analyzing large data sets. The code and data sets used in this research are available at the author's website [12]. Rec-I-DCM3 builds off of previous disc covering methods by recursively partitioning and merging subtrees for a set number of iterations. The algorithm proceeds through four main phases (see Figure 2):

- Decomposition
- Base Command
- Strict Consensus Merge
- Global Command

After Rec-I-DCM3 has read in the sequences and guide tree, it enters the *Decomposition* phase. In this phase, it recursively decomposes the guide tree into four subtrees until a user-specified limit is reached (see Figure 1).

When a tree is at least as small as the user-specified limit, a user-specified script is executed. Scripts are provided to allow PAUP* [13] and TNT [14] to be used for the base and global searches. The preparation of subtrees, executing the script and obtaining the results from the script are all part of the *Base Command* phase.

After the base command has completed on a set of four subtrees, the *Strict Consensus Merge* phase combines the subtrees. All subtrees are eventually merged together until a tree including all sequences is obtained.

At this point, a global search is performed using all sequences and a starting tree created from the consensus. This step is the *Global Command* phase. Rec-I-DCM3 repeats these phases until a user-specified number of iterations has been reached.

II. Analysis

Several experiments were performed in analyzing the behavior of Rec-I-DCM3. Although other data sets were analyzed, results for a single data set are presented here. This data set has 13,921 aligned 16s ribosomal Proteobacteria RNA sequences [15]. Tree scores reported in these experiments are calculated with Maximum Parsimony (as determined by PAUP). In the process of performing these experiments, several trees with scores better than those previously reported [10] were found. The best tree was found after using Rec-I-DCM3 for 56 hours and has a score of 240,919. Each of the experiments were executed on dedicated IBM X335 nodes (dual Pentium 2.4 GHz Xeon with 2 GB of memory), which are part of the Ira and Marylou Fulton Supercomputing Center at Brigham Young University [16].

A. Execution Phase Timing

As detailed in section I-A, Rec-I-DCM3 has four phases, Decomposition, Base Command, Strict Consensus Merge and Global Command. Figure 3 illustrates the amount of time spent in each phase for different permutations of the base and global commands. The Decomposition phase is not visible

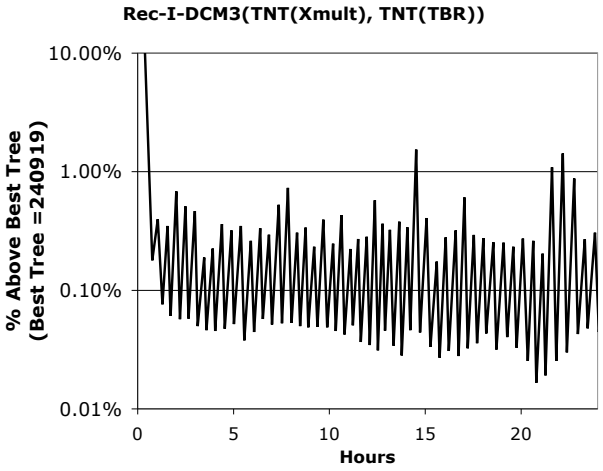

Fig. 4. Rec-I-DCM3(TNT(Xmult), TNT(TBR)) run illustrating typical Rec-I-DCM3 behavior.

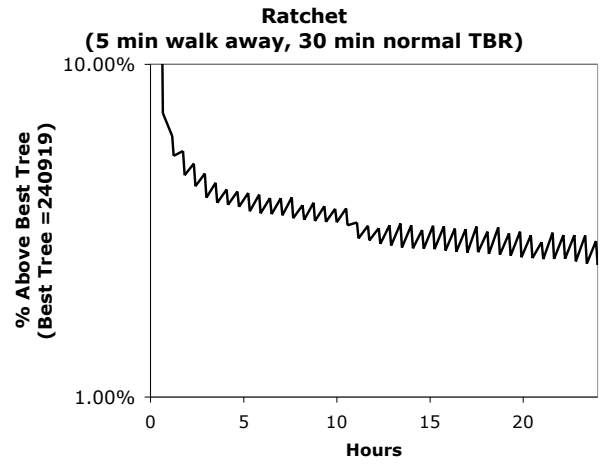

Fig. 5. Typical ratchet behavior with tree scores after the perturbation and refinement. (Note: y-axis is a logarithmic scale)

in the graph because it executed for less than a second. The Strict Consensus Merge phase is hardly visible, executing for only about a minute. It is interesting to note that with some configurations, the majority of the time is spent executing the global command. For future parallel implementations it is important to note that since the global command is performed with the whole data set, it must be executed on a single processor with current phylogenetic algorithms. While the recursive Decomposition, Base Command and Strict Consensus Merge phases naturally lend themselves to a parallel implementation on many processors, a parallel global command is necessary for efficient execution.

B. Tree Search

Figure 4 illustrates the typical behavior of Rec-I-DCM3. The graph is a representative run with TNT(Xmult) [14] as the base command and TNT(TBR) as the global command. TNT(Xmult) is a combination of the ratchet, drift [17], sectorial search and tree fusing [18]. TNT(TBR) is an implementation of the traditional tree bisection recombination method with some performance improvements. In this paper, the notation Rec-I-DCM3(x, y) refers to the Rec-I-DCM3 algorithm with x as the base command and y as the global command. The y-axis in the graph is the percentage above the best tree score. The following equation is used to define the percentage above best tree score:

$$\%\text{ above best tree score} = \frac{\text{tree score} - \text{best score}}{\text{best score}} \times 100\%$$

The y-axis is plotted on a logarithmic scale since generally the amount of effort to improve a tree score increases exponentially as the optimal tree score is approached. Scores of intermediate trees (the trees found after the first three phases have completed) as well as the score of the trees found at the end of each iteration are reported in the graph. Plotting the scores of the intermediate and iteration trees reveals that the first three phases almost always generate a worse tree, in terms of score. The first three phases only improved the tree score when the initial tree deviated more than 1% above the best tree score. The purpose of the Decomposition, Base Command and Strict Consensus Merge phases is not to find better trees, but serve to keep the global command from getting stuck in local minima. In this respect, DCM is similar to the ratchet algorithm [19] where column weights are randomly perturbed to avoid local minima. Figure 5 illustrates a typical ratchet run. The perturbation phase (walk away) randomly re-weights 25% of the columns and searches with TBR for five minutes, then with normal column weights for 30 minutes. The peaks in the graph are the tree scores after searching with perturbed column weights. The valleys correspond to the trees found after searching with the original weights. Running the ratchet algorithm on a set of sequences typically improves both the perturbed trees and the normally weighted trees.

Figure 6 shows the results of experiments with different permutations of base and global commands. The graph shows the first 24 hours of each run. Using TNT(Xmult) for the base command and TNT(TBR) for the global command yields the best tree scores in the shortest amount of time. This result agrees with other studies [9]. Additional analysis was performed to compare the results from the following algorithms:

- Rec-I-DCM3(TNT(Xmult), TNT(TBR))
- Iterations of TNT(Xmult)
- Iterations of TNT(TBR)

Figure 7 details the first 24 hours of these runs. While the other two runs continued past 24 hours, the execution of TNT(TBR) was terminated after more than 20 iterations without improvement of the tree score. Although TNT(Xmult) does not complete its first iteration until after six hours have elapsed, it returns a tree significantly better than TNT(TBR) found. At

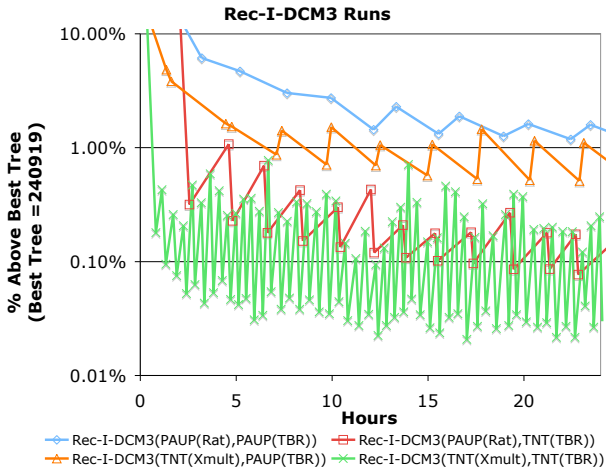

Fig. 6. Rec-I-DCM3 with different combinations of PAUP and TNT commands.

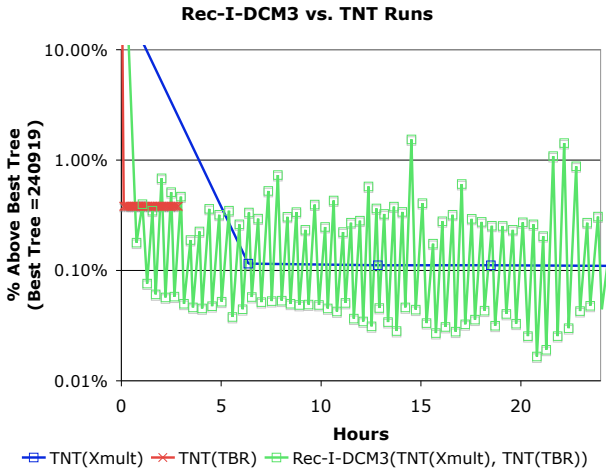

Fig. 7. Rec-I-DCM3 compared with TNT runs.

this same time, Rec-I-DCM3(TNT(Xmult), TNT(TBR)) has completed twelve iterations, and found better trees. Aside from runs of Rec-I-DCM3 with TNT(TBR) as the global command, TNT(Xmult) by itself returned trees with better scores than any other permutation of base and global commands or stand alone algorithms.

To separate the benefits of the first three phases, another set of experiments were set up with different base commands and no global command (see Figure 8). Although executing just the base command improved the tree scores, without a global refinement Rec-I-DCM3(TNT(Xmult), Nothing) and Rec-I-DCM3(TNT(TBR), Nothing) were not able to find trees with less than 1% difference in tree score. Furthermore, these two runs appear to be stuck in local minima. It is interesting that

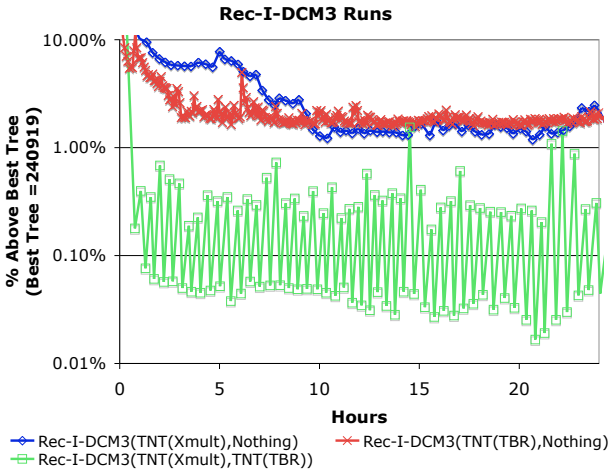

Fig. 8. Rec-I-DCM3 runs showing the contribution of the base command.

both TNT(Xmult) and TNT(TBR) performed far better than this, as illustrated in Figure 7.

III. Conclusion

Disk Covering Methods have been selected by the NSF funded CIPRes project as the basis for phylogenetic analysis with large data sets. These methods have been shown to be the most effective way of dealing with the extremely large search space involved when thousands of sequences are analyzed at the same time.

The following conclusions can be drawn from this research:

- The global phase of the algorithm must analyze all sequences in the data set. This phase often takes up the majority of the time in the Rec-I-DCM3 algorithm. While future parallel implementations of this algorithm can readily benefit from executing the base command for different subtrees on several processors, the global search must be performed on a single processor if current methods are used.
- The Rec-I-DCM3 algorithm is similar to the ratchet algorithm in that it produces an intermediate tree with a worse tree score, then refines this tree to obtain a better tree score.

Furthermore, for the large data set used in these experiments, a new best score of 240,919 was found.

IV. Future Work

Due to the inherent limitations imposed by searching on large data sets with current search algorithms, techniques to find better trees without a global search are being researched. Furthermore, parallel phylogenetic algorithms are being developed to increase the confidence of trees found and reduce search time.

REFERENCES

[1] A. Clark *et al.*, "Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase," *American Journal of Human Genetics*, vol. 63, pp. 595–612, 1998.

[2] C. Sing, M. Haviland, K. Zerba, and A. Templeton, "Application of cladistics to the analysis of genotype-phenotype relationships," *European Journal of Epidemiology*, vol. 8, pp. 3–9, 1992.

[3] K. Crandall, "Multiple interspecies transmissions of human and simian T-cell leukemia/lymphoma virus type I sequences," *Molecular Biology and Evolution*, vol. 13, pp. 115–131, 1996.

[4] R. DeSalle, "Molecular approaches to biogeographic analysis of Hawaiian Drosophilidae," in *Hawaiian Biogeography*, W. Wagner and V. Funk, Eds. Smithsonian Institute Press, 1995, pp. 72–89.

[5] D. Hillis, C. Miritz, and B. Mable, *Molecular Systematics*. Sinauer Associates Sunderland, 1996.

[6] D. Huson, S. Nettle, L. Parida, T. Warnow, and S. Yooseph, "A divide-and-conquer approach to tree reconstruction," in *Algorithms and Experiments (ALEX)*, Trent, Italy, Feb 1998, pp. 62–75.

[7] D. Huson, S. Nettle, and T. Warnow, "Disc-covering, a fast-converging method for phylogenetic tree reconstruction," *Computational Biology*, vol. 6, no. 3/4, pp. 369–386, 1999.

[8] D. Huson, L. Vawter, and T. Warnow, "Solving large scale phylogenetic problems using DCM-2," in *Proceedings of ISMB (Intelligent Systems for Molecular Biology)*, Saarbrucken, 1999.

[9] U. Roshan, B. M. E. Moret, T. Warnow, and T. Williams, "Rec-I-DCM3: A fast algorithmic technique for reconstructing large phylogenetic trees," in *Proceedings of IEEE Computational Systems Bioinformatics Conference*, August 2004, pp. 98–109.

[10] U. Roshan, "Algorithic techniques for improving the speed and accuracy of phylogenetic methods," Ph.D. dissertation, University of Texas at Austin, May 2004.

[11] "Cyberinfrastruce for phylogenetic research CIPRes project," http://www.phylo.org, March 2005.

[12] U. Roshan, http://www.cs.njit.edu/usman/, March 2005.

[13] D. L. Swofford, *PAUP*: Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4*, Sinauer Associates, Sunderland, Massachusetts, 2002.

[14] P. Goloboff, S. Farris, and K. Nixon, "TNT: Tree analysis using new technology," http://www.cladistics.com/webtnt.html, 2001.

[15] B. Maidak, J. Cole, T. Lilburn, C. P. Jr, P. Saxman, J. Stredwick, G. Garrity, B. Li, G. Olsen, S. Pramanik, T. Schmidt, and J. Tiedje, "The RDP (ribosomal database project) continues," *Nucleic Acids Research*, vol. 28, pp. 173–174, 2000.

[16] "Ira and Mary Lou Fulton Supercomputing Center," http://marylou.byu.edu, March 2005.

[17] P. Gologoff, "Analyzing large data sets in reasonable times: solutions for composite optima," *Cladistics*, vol. 15, pp. 415–428, 1999.

[18] P. Hovenkamp, *TNT Manual*, June 2004.

[19] K. Nixon, "The parsimony ratchet, a new method for rapid parsimony analysis," *Cladistics*, vol. 15, pp. 407–414, 1999.