

# PRec-I-DCM3: A Parallel Framework for Fast and Accurate Large Scale Phylogeny Reconstruction <sup>\*</sup> <sup>†</sup> <sup>‡</sup>

Cristian Coarfa  
Rice University  
Houston, TX 77005, USA  
ccristi@cs.rice.edu

Yuri Dotsenko  
Rice University  
Houston, TX 77005, USA  
dotsenko@cs.rice.edu

John Mellor-Crummey  
Rice University  
Houston, TX 77005, USA  
johnmc@cs.rice.edu

Luay Nakhleh  
Rice University  
Houston, TX 77005, USA  
nakhleh@cs.rice.edu

Usman Roshan  
New Jersey Institute of Technology  
Newark, NJ 07102, USA  
usman@cs.njit.edu

## Abstract

*Accurate reconstruction of phylogenetic trees very often involves solving hard optimization problems, particularly the maximum parsimony (MP) and maximum likelihood (ML) problems. Various heuristics have been devised for solving these two problems; however, they obtain good results within reasonable time only on small datasets. This has been a major impediment for large-scale phylogeny reconstruction, particularly for the effort to assemble the Tree of Life—the evolutionary relationship of all organisms on earth. Roshan et al. recently introduced Rec-I-DCM3, an efficient and accurate meta-method for solving the MP problem on large datasets of up to 14,000 taxa. Nonetheless, a drastic improvement in Rec-I-DCM3’s performance is still needed in order to achieve similar (or better) accuracy on datasets at the scale of the Tree of Life. In this paper, we improve the performance of Rec-I-DCM3 via parallelization. Experimental results demonstrate that our parallel method, PRec-I-DCM3, achieves significant improvements, both in speed and accuracy, over its sequential counterpart.*

## 1. Introduction

Phylogenies play a major role in representing the evolutionary relationships among groups of taxa. Their pervasiveness has led biologists, mathematicians, and computer scientists to develop a wide array of methods for their reconstruction. One of the open problems facing biology today is reconstruction of the Tree of Life—the evolutionary history of all organisms on earth. Fundamental to this reconstruction is the ability to produce, within reasonable time constraints, accurate phylogenies for large datasets (tens to hundreds of thousands of taxa), since the Tree of Life itself is estimated to contain tens to hundreds of millions of taxa. The most commonly used approaches to phylogeny reconstruction are heuristics for two hard optimization problems, maximum parsimony (MP) and maximum likelihood (ML). However, despite decades of research and algorithm development, acceptably accurate analyses that run within a few days of computation on one processor are not currently possible for much beyond a few thousand taxa for MP and a few hundred taxa for ML—nor is it clear that increases in the computing power will enable adequate analysis of larger datasets, as the accuracy of the heuristics steadily decreases with increasing size of datasets. Polynomial-time algorithms do exist (Neighbor-Joining [9] and UPGMA [5] are the best known examples), but many experimental studies have shown that such trees are not as accurate as those produced by MP or ML analy-

---

\* Cristian Coarfa and Yuri Dotsenko contributed equally to this work.

† This work was supported in part by the Department of Energy under Grant DE-FC03-01ER25504/A000. The computations were performed on an Itanium cluster purchased with support from the NSF under Grant EIA-0216467, Intel, and Hewlett Packard.

‡ The authors would like to thank Erion Plaku for helpful discussions, and Derek Ruths for providing us with the code for computing the Robinson-Foulds distance between trees.

ses. Because MP approaches are more accurate than polynomial-time methods and are significantly faster than ML analyses, which are sometimes more accurate, the majority of published phylogenies to date have been derived using MP-based heuristics [10].

In [8], Roshan *et al.* presented a new technique that makes it possible to reach an acceptable level of accuracy on datasets of large size—indeed, of sizes at least one order of magnitude larger than could be analyzed before. Their technique, called *Recursive-Iterative DCM3* (Rec-I-DCM3), employs a divide-and-conquer strategy that combines recursion and iteration with a new variant of the *Disk-Covering Method (DCM)* to find highly accurate trees quickly. Roshan *et al.* showed that Rec-I-DCM3 convincingly outperformed TNT [3]—the best implemented MP heuristic—often by orders of magnitude, on all datasets and at all times during the time period (usually 24 hours) allotted for computation.

If it is to be useful for analyzing large datasets at the scale of the Tree of Life within reasonable time limits and with high accuracy, the performance of Rec-I-DCM3 must be improved by orders of magnitude. In this paper, we address the problem of large scale phylogenetic tree reconstruction by building on the success of Rec-I-DCM3. We have designed and implemented a parallel version of the method, called PRec-I-DCM3. We have tested PRec-I-DCM3 on the two largest biological datasets used in [8]. Our results show a drastic improvement in the performance of the method. For example, on the largest dataset used by Roshan *et al.*, Rec-I-DCM3 took about 13 hours to find the MP tree found by PRec-I-DCM3 within less than three hours. Further, the parsimony scores of trees computed by PRec-I-DCM3 are consistently better than those computed by Rec-I-DCM3 within the same amount of time.

## 2. Disk-Covering Methods

*Disk-Covering Methods (DCMs)* (e.g., [11]) are a family of divide-and-conquer methods designed to “boost” the performance of existing phylogenetic reconstruction methods. All DCMs proceed in four major phases: (i) decomposing the dataset, (ii) solving the subproblems, (iii) merging the subproblems, and (iv) refining the resulting tree.

The Rec-I-DCM3 method [8] takes as input the set  $S = \{s_1, \dots, s_n\}$  of  $n$  aligned biomolecular sequences, the chosen base method, and a starting tree  $T$  used as a guide tree to direct the search. In [8], the authors used

TNT (with default settings) as the base method, since it is the hardest to improve (in comparison, the PAUP\* implementation of the parsimony ratchet [1] is easier to improve).

The Rec-I-DCM3 method is aimed mainly at improving the performance of *maximum parsimony* heuristics. The parsimony criterion is a reflection of Occam’s razor: the tree with the minimum number of mutations along its branches best explains the data. The *Maximum Parsimony (MP)* problem seeks the phylogenetic tree with the minimum amount of change along its branches. The MP problem is NP-hard [2].

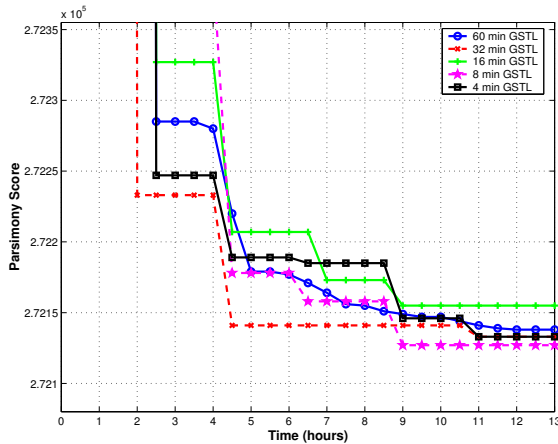
## 3. Parallel Rec-I-DCM3

As described in section 2, Rec-I-DCM3 is a divide-and-conquer algorithm, which makes it a natural candidate for parallelization. For the datasets we studied (up to 14000 taxa), the problem fits into memory, which simplifies the implementation. A typical problem decomposition contains the main problem, composite subproblems (those which can be decomposed further), and leaf subproblems (those at the final level of decomposition).

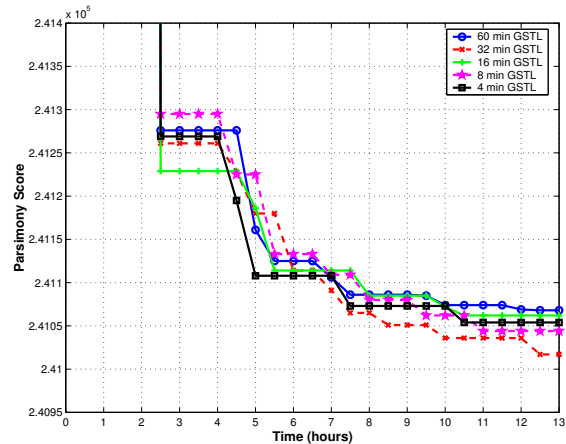
The natural implementation for PRec-I-DCM3 is to use task-parallelism with a master-slave model. The master node maintains a database of subproblems: subproblems available for solving (“available”), already solved subproblems (“solved”), and subproblems currently being solved by a worker (“active”). During its lifetime, a subproblem changes states from “available” to “active” to “solved”. The master coordinates the distributed computation and ensures that the system is in a consistent state throughout the computation.

At the beginning of a PRec-I-DCM3 iteration, the master performs a one-level decomposition of the main problem, using the current guide tree (see [8] for details regarding the use of a guide tree). Among the resultant subproblems, there are usually both leaf subproblems and composite ones. Next, it dispatches available subproblems, in decreasing order of their sizes, to the idle workers. The rationale for distributing large problem first is that they will take longer to solve. Once a problem is dispatched, its state is changed from “available” to “active”.

Worker processes wait for subproblems from the master. If the problem that a worker receives is a leaf subproblem, then the worker invokes a standalone solver, such as TNT or PAUP\*, and then returns the resulting subtree to the master. The solver runs without a time limit, because the leaf subproblems are small (usually not exceeding 2000 taxa). If the problem received is



(a) European RNA dataset



(b) RDPII dataset

**Figure 1. Results obtained by Rec-I-DCM3 on the European RNA and RDPII datasets with maximum subproblem size of 2000 taxa and different GSTL values.**

a composite subproblem, the worker decomposes it further into subproblems. It selects the largest subproblem among these to perform additional work on it and returns the remaining subproblems to the master.

When the master receives the solution for a leaf subproblem, it changes its state from “active” to “solved”. It checks if all of the leaf’s sibling subproblems have been solved; if so, then it sends all of these subproblems to a worker for merging.

When the master receives the decomposition of a composite subproblem, it adds all of the child subproblems to the problem database, with the state “available”, and marks the subproblem kept by the worker as “active”.

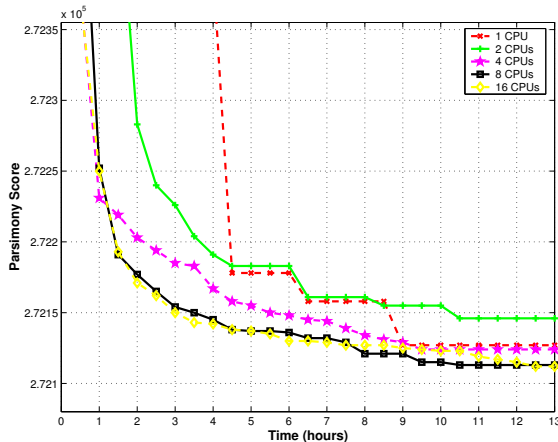
When a worker receives a command to merge a composite subproblem, it receives the solutions for the child subproblems from the master and applies the strict consensus merging [8], followed by a random refinement of the resulting tree. Finally, the solution is sent to the master.

When all the subproblems of the main problem are solved, the master merges their solution trees, then signals the workers to perform the random refinement (making the tree binary) followed by the search phase on the tree. These random refinements are performed independently by each worker. Since each worker process has an independent random number generator, the refinements usually result in different trees. After refining the merged trees, workers

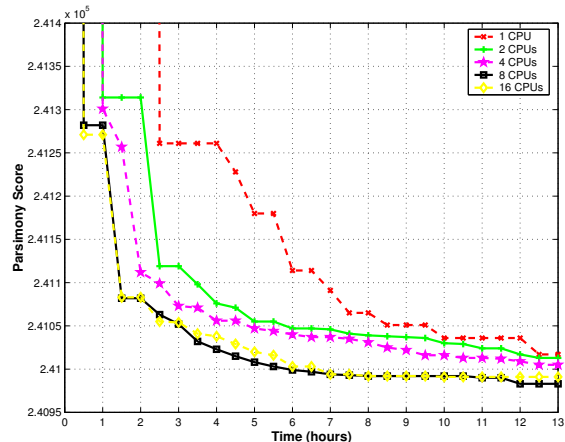
perform the *global search* phase: they invoke the standalone solver on the refined trees to search for trees with better parsimony scores. The time of the global search is limited to a fixed value (referred to as the *Global Search Time Limit*, or *GSTL*), which is specified at program launch. The limit is necessary because the full-size problem is large, up to 14000 taxa. After finishing the global search phase, each worker sends its resultant parsimony scores to the master, which, in turn, selects the minimum score and retrieves the corresponding tree from the worker. This tree is used as the guide tree for the following iteration.

A significant advantage of PRec-I-DCM3 over Rec-I-DCM3 is that the former is able to run several instances of the global search phase in parallel. These instances start from different points in the tree space, potentially leading to a wider coverage of the tree search space and a new option for escaping local optima.

The current master-slave scheme implementation is a prototype. For very large datasets (20000+ taxa), the master can become a bottleneck, flooded with incoming and outgoing communication traffic. A distributed master scheme will solve this problem by distributing the database of subproblems onto several nodes. An appealing technology to implement this solution is emerging global address space languages, such as Co-Array Fortran [6]. They provide the abstraction of a global address space and support one-sided communication as part of



(a) European RNA dataset



(b) RDPII dataset

**Figure 2. Results obtained by  $\text{Rec-I-DCM3}$  and  $\text{PRec-I-DCM3}$  on the European RNA dataset with maximum subproblem size of 2000 taxa and GSTL of 8 minutes, and on the RDPII dataset with maximum subproblem size of 2000 taxa and GSTL of 32 minutes. The 1-CPU curves correspond to the  $\text{Rec-I-DCM3}$ , whereas the other curves correspond to  $\text{PRec-I-DCM3}$  using different numbers of CPUs.**

the language. This makes them well-suited for developing a distributed subproblem database.

#### 4. Experimental Settings and Results

The platform used for experiments was a cluster of 92 HP zx6000 workstations with a Myrinet 2000 interconnect. Each workstation contains two 900MHz Intel Itanium 2 processors with 32KB/256KB/1.5MB of L1/L2/L3 cache, 4GB of RAM, and the HP zx1 chipset. Each node is running the Linux operating system (kernel version 2.4.18-e plus patches). We used Intel’s C/C++ compiler version 8.1 for Itanium. Each node used the UNIX `random` random number generator initialized with a seed read from node’s `/dev/random` at the beginning of a run.

We ran both  $\text{Rec-I-DCM3}$  and  $\text{PRec-I-DCM3}$  on the two largest datasets used in [8]: European RNA, a dataset of 11,361 aligned small subunit ribosomal Bacteria RNA sequences (1,360 sites) [12], and RDPII, a dataset of 13,921 aligned 16s ribosomal Proteobacteria RNA sequences (1,359 sites) [4]. We report the average results of the methods over five runs on the two datasets.

We investigated two main questions: (1) what is the optimal choice of GSTL and subproblem size that yields the best results of  $\text{Rec-I-DCM3}$ ? (2) Using the optimal

choice of parameters, how does  $\text{PRec-I-DCM3}$  perform compared to  $\text{Rec-I-DCM3}$ ?

To answer the first question, we ran  $\text{Rec-I-DCM3}$  on the European RNA and RDPII datasets for 13 hours, and recorded the average best scores obtained by the method for various subproblem sizes and GSTL values.

We determined that a maximum subproblem size of 2000 taxa yields the best results on both datasets, under the conditions of our experiments. However, a GSTL of 8 minutes gave the best results on the European RNA dataset, while a GSTL of 32 minutes gave the best results on the RDPII dataset.

Using the maximum subproblem size of 2000 taxa, we investigated the behavior of  $\text{Rec-I-DCM3}$  for different values of GSTL for the duration of 13-hour runs. Figures 1(a) and (b) show that a GSTL of 8 minutes becomes consistently the optimal choice on the European RNA dataset after about 8 hours and 40 minutes, whereas a GSTL of 32 minutes becomes consistently the optimal choice on the RDPII dataset after about 6 hours and 30 minutes.

These results demonstrate that different datasets may require different parameter settings of  $\text{Rec-I-DCM3}$  to achieve the best performance. We expect that the choice of these parameters depends on the quality (in terms of parsimony score and topology) of the tree used as the start point in each iteration, as well as the evolutionary

diameter<sup>1</sup> of the dataset. We will investigate this in future work.

After determining the optimal choice of maximal subproblem size and GSTL values, we used these values for `PRec-I-DCM3` and compared its performance to that of `Rec-I-DCM3` under the same settings. To compare the performance of `PRec-I-DCM3` and `Rec-I-DCM3`, we ran both methods on the two datasets for 13 hours, using a maximal subproblem size of 2000 taxa, and GSTL values of 8 and 32 minutes for the European RNA and RDPII datasets, respectively. We plotted the average parsimony score obtained by the two methods as a function of time; the results are shown in Figure 2.

Figure 2(a) shows that, on the European RNA dataset, `PRec-I-DCM3` consistently outperforms `Rec-I-DCM3`, with the exception of the 2-CPU case. The figure shows that the best parsimony score computed by `Rec-I-DCM3` after 6 and a half hours is computed by `PRec-I-DCM3` after only two and a half hours using 8 or 16 workers. Further, the best parsimony score computed by `Rec-I-DCM3` after a complete run of 13 hours is computed by `PRec-I-DCM3` (using 8 and 16 workers) after only 7 and a half hours. Finally, despite a seemingly small difference in the parsimony scores computed by the two methods after 13 hours, trees computed by the methods differ in about 25% of their internal edges, according to the RF[7] metric (detailed comparisons are omitted due to space limitations).

More dramatic improvements were observed on the RDPII dataset, as Figure 2(b) demonstrates. On this dataset, the performance of `PRec-I-DCM3` is consistently better than that of `Rec-I-DCM3`, regardless of the number of workers used for the `PRec-I-DCM3` execution. The best performance of `PRec-I-DCM3` on this dataset is achieved using 8 and 16 worker CPUs, with a slight edge for the 8-CPU implementation after 11 hours. Notice that the best parsimony score computed by `Rec-I-DCM3` after the complete run of 13 hours is obtained by `PRec-I-DCM3` using 8 CPUs after only 4 hours. Our experiments indicate that the difference between the parsimony scores obtained by the two methods after 13 hours translates into a 40% difference in the topologies (specifically, numbers of internal edges) of the trees computed by the two methods, according to the RF measure.

---

<sup>1</sup> The evolutionary diameter of a dataset is defined as the maximum number of changes between any two taxa in the dataset.

## References

- [1] O. Bininda-Emonds. Ratchet implementation in PAUP\*4.0b10, 2003. Available from [www.tierzucht.tum.de:8080/WWW/Homepages/Bininda-Emonds](http://www.tierzucht.tum.de:8080/WWW/Homepages/Bininda-Emonds).
- [2] L. Foulds and R. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.
- [3] P. Goloboff. Analyzing large data sets in reasonable times: solution for composite optima. *Cladistics*, 15:415–428, 1999.
- [4] 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*, 28:173–174, 2000.
- [5] C. Michener and R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.
- [6] R. W. Numrich and J. K. Reid. Co-Array Fortran for parallel programming. Technical Report RAL-TR-1998-060, Rutheford Appleton Laboratory, August 1998.
- [7] D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.
- [8] U. Roshan, M. E. Moret, T. L. Williams, and T. Warnow. Rec-I-DCM3: A fast algorithmic technique for reconstructing large phylogenetic trees. In *Proceedings of the IEEE Computational Systems Bioinformatics conference (CSB) 2004*, 2004.
- [9] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.
- [10] M. Sanderson, B. Baldwin, G. Bharathan, C. Campbell, D. Ferguson, J. Porter, C. V. Dohlen, M. Wojciechowski, and M. Donoghue. The growth of phylogenetic information and the need for a phylogenetic database. *Systematic Biology*, 42:562–568, 1993.
- [11] 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.
- [12] 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.