PRec-I-DCM3: a parallel framework for fast and accurate large-scale phylogeny reconstruction

Yuri Dotsenko*, Cristian Coarfa, Luay Nakhleh and John Mellor-Crummey

Department of Computer Science,
Rice University, 6100 Main Street,
Houston TX 77005, USA
E-mail: dotsenko@cs.rice.edu E-mail: ccristi@cs.rice.edu
E-mail: nakhleh@cs.rice.edu E-mail: johnmc@cs.rice.edu
*Corresponding author

Usman Roshan

Department of Computer Science,
New Jersey Institute of Technology,
GITC 4400, University Heights,
Newark NJ 07102, USA
E-mail: usman@cs.njit.edu

Abstract: Accurate reconstruction of phylogenetic trees often involves solving hard optimisation problems, particularly the Maximum Parsimony (MP) and Maximum Likelihood (ML) problems. Various heuristics yield good results for these problems within reasonable time only on small datasets. This is a major impediment for large-scale phylogeny reconstruction. Roshan et al. introduced Rec-I-DCM3, an efficient and accurate meta-method for solving the MP problem on large datasets of up to 14,000 taxa. We improve the performance of Rec-I-DCM3 via parallelisation. The experiments demonstrate that our parallel method, PRec-I-DCM3, achieves significant improvements, both in speed and accuracy, over its sequential counterpart.

Keywords: phylogeny; Maximum Parsimony (MP); Disk-Covering Method (DCM); DCM3; Rec-I-DCM3; PRec-I-DCM3; parallel computing; scalability; bioinformatics research and applications.

Reference to this paper should be made as follows: Dotsenko, Y., Coarfa, C., Nakhleh, L., Mellor-Crummey, J. and Roshan, U. (2006) 'PRec-I-DCM3: a parallel framework for fast and accurate large-scale phylogeny reconstruction', *Int. J. Bioinformatics Research and Applications*, Vol. 2, No. 4, pp.407–419.

Biographical notes: Yuri Dotsenko is a PhD student in the Department of Computer Science at Rice University, Houston, TX, where he also received his MS Degree. His research interests include compiler technology for parallel computing, distributed computing and bioinformatics.

Cristian Coarfa is a PhD student in the Department of Computer Science at Rice University, Houston, TX, where he also received his MS Degree. His research interests include parallel and distributed computing, bioinformatics and advanced compiler technology.

Luay Nakhleh received his PhD Degree in Computer Science in 2004 from the University of Texas, Austin, TX. His Doctoral Dissertation was about Phylogenetic Networks in Biology and Historical Linguistics. He joined the Department of Computer Science at Rice University as an Assistant Professor in July 2004. His research interests include phylogenetics, bacterial genomics and computational gene finding.

John Mellor-Crummey received his PhD in Computer Science from the University of Rochester in 1989. He joined Rice University in 1989, where he is now an Associate Professor in the Department of Computer Science. Since 2002, he has been Deputy Director of Rice's Center for High Performance Software Research. His research focuses on compiler, tool and run-time library support for high performance computing.

Usman Roshan received his PhD Degree in Computer Science in 2004 from the University of Texas at Austin. His Dissertation topic was Algorithms for Constructing Phylogenies on Large Datasets. He joined the Computer Science Department at the New Jersey Institute of Technology in August 2004 as an Assistant Professor. His research interests are phylogenetics, alignment, comparative approaches in bioinformatics and high performance computing.

1 Introduction

Phylogenies play a major role in representing the evolutionary relationships among groups of taxa. Their importance 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 optimisation problems, MP and 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 (Neighbour-Joining (Saitou and Nei, 1987) and UPGMA (Michener and Sokal, 1957) 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 analyses. 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 (Sanderson et al., 1993).

Whereas 90–95% accuracy is often considered excellent in heuristics for hard optimisation problems, heuristics used in phylogenetic reconstruction must be much more accurate: Williams and Moret found that solutions to MP that had an error rate larger than 0.01% (i.e., whose length exceeded the optimal length by more than 0.01%) produced

topologically poor estimates of the true tree (Williams et al., 2004). Thus, heuristics for MP need at least 99.99% accuracy (and probably significantly more on very large datasets) in order to produce topologically accurate trees. Obtaining this level of accuracy while running within a reasonable time presents a stiff challenge to algorithm developers.

In Roshan et al. (2004b) 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 analysed 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 *DCM* to compute highly accurate trees quickly. Rec-I-DCM3 uses iteration for escaping local optima, the divide-and-conquer approach of the DCMs to reduce problem size, and recursion to enable further localisation and reduction in problem size. A Rec-I-DCM3 search not only dramatically reduces the size of the explored tree space, but also finds a larger fraction of MP trees with better scores than other methods. Roshan et al. demonstrated the power of Rec-I-DCM3 on ten large bio-molecular sequence datasets, each containing more than 1,000 sequences (half contain over 6,000 sequences and the largest contains almost 14,000 sequences). Their study showed that Rec-I-DCM3 convincingly outperformed `TNT` (Goloboff, 1999) – 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 analysing 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 investigated, through experiments on biological datasets, the best choice of parameters whose values affect the performance of Rec-I-DCM3. Further, we have designed and implemented a parallel version of the method, called PRec-I-DCM3. We have compared PRec-I-DCM3 and Rec-I-DCM3 on the two largest biological datasets used in Roshan et al. (2004b). 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 Maximum Parsimony (MP)

The parsimony criterion is but a reflection of Occam's razor: the tree with the minimum number of mutations along its branches best explains the data. In this section, we review the formal definition of the MP problem and the latest heuristics for solving it.

Let S be a set of sequences, each of length n, over a fixed alphabet Σ. Let T be a tree leaf-labelled by the set S and with internal nodes labelled by sequences of length n over Σ. The *length* (or *parsimony score)* of T with this labelling is the sum, over all the edges, of the Hamming distances between the labels at the endpoints of the edge. (The Hamming distance between two strings of equal length is the number of positions in which the two strings differ.) Thus the length of a tree is also the total number of point mutations along the edges of the tree. The *MP* problem seeks the tree T leaf-labelled by S with the minimum length. While MP is NP-hard (Foulds and Graham, 1982),

constructing the optimal labelling of the internal nodes of a fixed tree T can be done in polynomial time (Fitch, 1971).

2.1 Iterative improvement methods

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 in order to find trees with a better score. The most commonly used local move is called *Tree-Bisection and Reconnection (TBR)* (Maddison, 1991). In TBR, 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.

The *Parsimony Ratchet* (Nixon, 1999) is an iterative technique that combines TBR with an interesting approach to move out of local optima. When a local optimum has been reached, i.e., when no further improvement can be made through a TBR move, the input data are modified by randomly doubling p% of the sites to produce new sequences that are $1 + p$ times longer than the original input sequences. (Typically, p is 0.25.) Iterative improvement with TBR is then attempted on the new data. When this new search reaches a local optimum, the additional sites are removed (reverting to the original sequence length) and iterative improvement is resumed from this new configuration. The parsimony ratchet is implemented in two software packages, `TNT` (Goloboff, 1999) and `PAUP*` (Swofford, 2002). `TNT` provides a faster implementation; neither package is publicly available.

2.2 Disk-Covering Methods (DCMs)

DCMs (Huson et al., 1999a, 1999b; Nakhleh et al., 2001; Roshan et al., 2004a; Warnow et al., 2001) 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:

- decomposing the dataset
- solving the subproblems
- merging the subproblems
- refining the resulting tree.

Variants of DCMs come from different decomposition methods – the last three phases are unaffected. The first DCM (Huson et al., 1999a), also called DCM1, was designed for use with distance-based methods and has provable theoretical guarantees about the sequence length required to reconstruct the true tree with high probability under Markov models of evolution (Warnow et al., 2001). The second DCM (Huson et al., 1999b), also called DCM2, was designed to speed up heuristic searches for MP trees.

3 Rec-I-DCM3

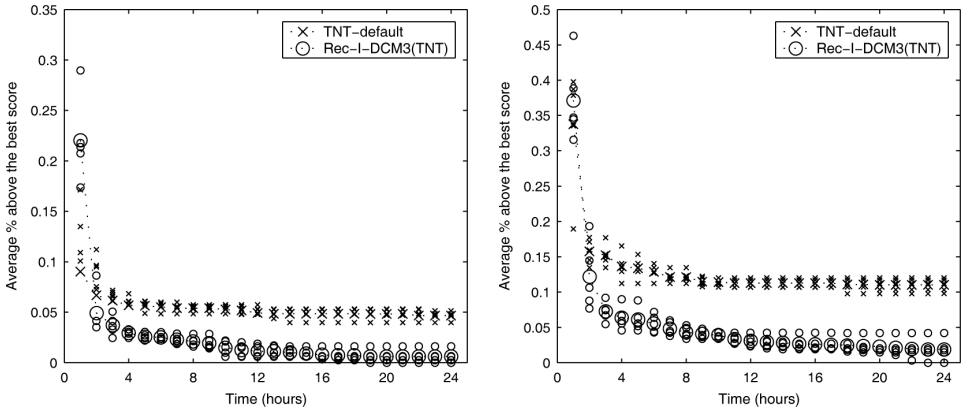
DCM1 can be viewed, in rough terms, as attempting to produce overlapping clusters of taxa to minimise the intra-cluster diameter; it produces good subproblems (small enough in size), but the structure induced by the decomposition is often poor. DCM2 computes a fixed structure (a graph separator) to overcome that drawback, but the resulting subproblems tend to be too large. Moreover, both DCM1 and DCM2 operate solely from the the matrix of estimated pairwise distances, so that they can produce only one (up to tiebreaking) decomposition. In contrast, DCM3 uses a dynamically updated *guide tree* (in practice, the current estimate of the phylogeny) to direct the decomposition – so that DCM3 will produce different decompositions for different guide trees. This feature allows to focus the search on the best parts of the search space and is at the heart of the iterative use of the decomposition: roughly speaking, the iteration in Rec-I-DCM3 consists of successive refinements of the guide tree. Thanks to the guide tree, DCM3 also produces smaller subproblems than DCM2: the guide tree provides the decomposition structure, but does so in a manner responsive to the phylogenetic estimation process. Finally, DCM3 was designed to be much faster than either DCM1 or DCM2 in producing the decompositions (mostly by not insisting on their optimality), since previous experiments had shown that dataset decomposition contributed the most to the running time of DCM2.

Roshan et al. designed DCM3 in part to avoid producing large subsets, as DCM2 is prone to do (Roshan et al., 2004b). Yet, of course, the subproblems produced from a very large dataset remain too large for immediate solution by a base method – a phylogenetic tree reconstruction method of choice. Hence they used DCM3 recursively, producing smaller and smaller subproblems until every subproblem was small enough to be solved directly. In Roshan et al. (2004b) showed that DCM3 produced subproblems of sizes bounded by about half the initial subproblem size and much smaller than those produced by DCM2. (Rec-I-DCM3 in that series of tests was set up to recurse until the size of each subproblem was at most one quarter of the original problem size.)

Once the dataset is decomposed into overlapping subsets $A_1, A_2, \ldots, A_m$ ($m \leq 4$ is typical), subtrees are constructed for each subset, A_i, using the chosen base method and then combined using the Strict Consensus Merger (Huson et al., 1999a, 1999b) to produce a tree on the combined dataset.

The Rec-I-DCM3 method (Roshan et al., 2004b) takes as input the set $S = \{s_1, \ldots, s_n\}$ of n aligned bio-molecular sequences, the chosen base method, and a starting tree T. In Roshan et al. (2004b), the authors used `TNT` (with default settings) as the base method, since it is the hardest to outperform (in comparison, the results of the `PAUP*` implementation of the parsimony ratchet (Bininda-Emonds, 2003) are easier to improve upon). The Rec-I-DCM3 method produces smaller subproblems by recursively applying the centroid-edge decomposition until each subproblem is of size at most k. The subtrees are then computed, merged and resolved bottom-up, using random resolution, to obtain a binary tree on the full dataset. These steps are repeated for a specified number of iterations. Figure 1 demonstrates the significant improvement over `TNT` gained by employing the Rec-I-DCM3 booster to the method.

Figure 1 (a) Average MP scores of TNT and Rec-I-DCM3(TNT) on the European RNA dataset, given as the percentage above the best score. Shown are the data points of five runs of both methods indicated by small symbols. After the fourth hour there is no overlap of points and the variances of both the methods are low. Note: the vertical range varies across the datasets and (b) average MP scores of TNT and Rec-I-DCM3(TNT) on the RDPII dataset, given as the percentage above the best score. Also shown are the data points of five runs of both methods indicated by small symbols. Note that the variances are very low and after the third hour there is no overlap of points

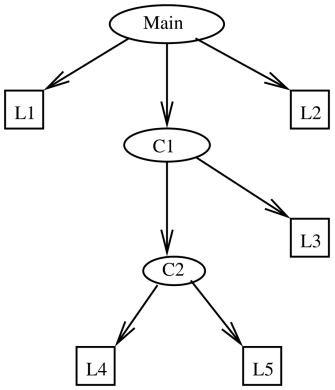


4 Parallel Rec-I-DCM3

As described in Section 3, Rec-I-DCM3 is a divide-and-conquer algorithm, which makes it a natural candidate for parallelisation. For the datasets we studied (up to 14000 taxa), the problem fits into memory, which simplifies the implementation.

In Figure 2, we present a typical problem decomposition induced by Rec-I-DCM3; the decomposition contains the main problem, composite subproblems (those which can be decomposed further: C1, C2), and leaf subproblems (those at the final level of decomposition: L1–L5).

Figure 2 The Rec-I-DCM3 decomposition: main problem, composite subproblems (C1–C2) and leaf subproblems (L1–L5)



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 Roshan et al. (2004b) for details regarding the use of a guide tree). Among the resultant subproblems, there are usually both leaf and composite subproblems. Next, it dispatches available subproblems, in decreasing order of their sizes, to the idle workers. The rationale for distributing larges problem first is that they are likely to 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 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 (Huson et al., 1999a, 1999b; Roshan et al., 2004b), 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, 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 (Numrich and Reid, 1998). They provide the abstraction of a global address space and support one-sided communication as part of the language. This makes them well-suited for developing a distributed subproblem database.

5 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 900 MHz Intel Itanium 2 processors with 32 KB/256 KB/1.5 MB of L1/L2/L3 cache, 4 GB 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. Since Rec-I-DCM3 uses strict consensus merging with random refinement, a good random number generator is essential to obtain credible results. Each node used the UNIX `random` random number generator initialised with a seed read from the node's/`dev`/`random` at the beginning of a run.

We ran both Rec-I-DCM3 and PRec-I-DCM3 on the two largest datasets used in Roshan et al. (2004b): European RNA, a dataset of 11,361 aligned small subunit ribosomal Bacteria RNA sequences (1,360 sites) (Wuyts et al., 2002), and RDPII, a dataset of 13,921 aligned 16 s ribosomal Proteobacteria RNA sequences (1,359 sites) (Maidak et al., 2000). We report the average results of the methods over five runs on the two datasets.

We investigated two main questions:

- what is the optimal choice of GSTL and subproblem size that yields the best results of Rec-I-DCM3?
- using the optimal choice of parameters, how does PRec-I-DCM3 perform compared to Rec-I-DCM3?

To answer the first question, we ran 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.

Tables 1 and 2 show the average best scores obtained by Rec-I-DCM3 on the European RNA and RDPII datasets, respectively. 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.

Table 1 Average best scores obtained by Rec-I-DCM3 on the European RNA dataset, with different GSTL values and subproblem sizes

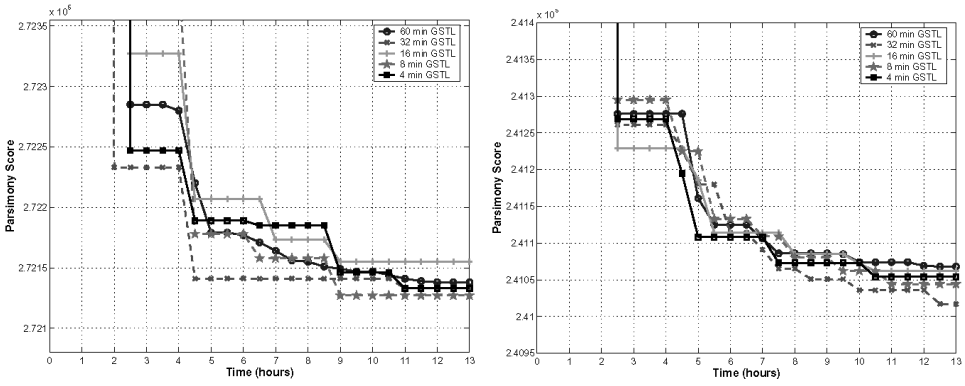
Max. subproblem size\GSTL	*4* min	*8* min	*16* min	*32* min	*60* min
500	273688	272326	272209	272160	272163
1000	272865	272194	272158	272163	272145
2000	272133	*272127*	272155	272133	272138
4000	272151	272146	272209	272154	272155

Table 2 Average best scores obtained by Rec-I-DCM3 on the RDPII dataset, with different GSTL values and subproblem sizes

Max. subproblem size\GSTL	*4* min	*8* min	*16* min	*32* min	*60* min
500	243545	242005	241131	241093	241088
1000	242529	241275	241140	241069	241042
2000	241054	241044	241062	*241017*	241068
4000	241135	241154	241118	241131	241097

Using the maximum subproblem size of 2000 taxa, we investigated the behaviour of Rec-I-DCM3 for different values of GSTL for the duration of 13-hour runs. Figures 3(a) and 3(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.

Figure 3 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) European RNA dataset and (b) RDPII dataset



(a) (b)

These results demonstrate that different datasets may require different parameter settings of 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[1] of the dataset. We will investigate this in future work.

After determining the optimal choice of maximal subproblem size and GSTL, we used these values for PRec-I-DCM3 and evaluated its performance against 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, and the topological difference between the best trees computed by the two methods as computed by the Robinson-Foulds (RF) metric (Robinson and Foulds, 1981) of topological tree difference. We now briefly review the RF metric.

Let T be an unrooted tree leaf-labelled by a set S of taxa. An edge $e = (u, v)$ in T defines a bipartition of S (the set of all leaves on one side of the edge, and the set of all other leaves). Let $C(T)$ be the set of bipartitions defined by all edges in tree T. The RF measure between two trees T and T' is defined as

$$RF(T,T') = \frac{(|C(T) - C(T')|/|C(T)|) + (|C(T') - C(T)|/|C(T')|)}{2}.$$

Figure 4(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 best parsimony score computed by Rec-I-DCM3 after 6 and a half hours is computed by PRec-I-DCM3 after only 2 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, Figure 5(a) shows that the actual trees computed by the methods differ in about 25% of their internal edges, according to the RF metric. This result shows a significant difference between the trees computed by the two methods on the European RNA dataset.

Figure 4 Results obtained by Rec-I-DCM3 and 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 Rec-I-DCM3, whereas the other curves correspond to PRec-I-DCM3 using different numbers of CPUs: (a) European RNA dataset and (b) RDPII dataset

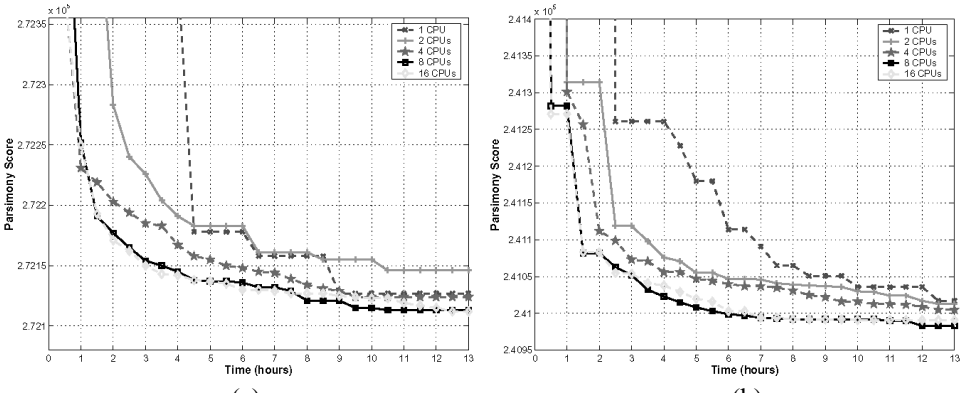
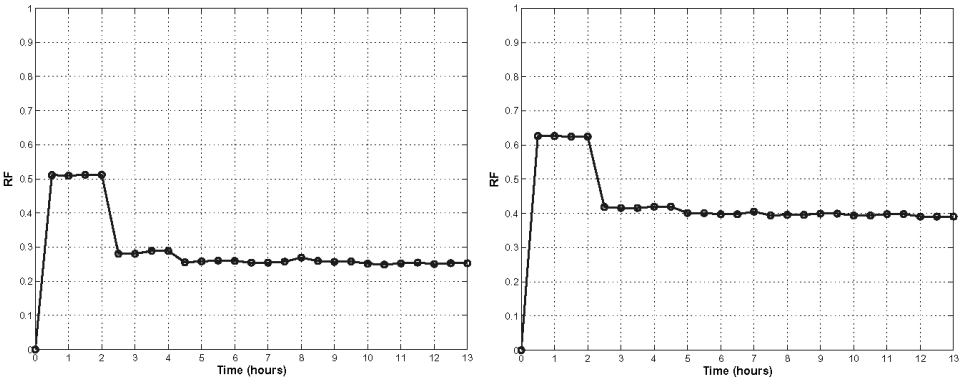


Figure 5 RF values between the best trees obtained by Rec-I-DCM3 on and PRec-I-DCM3 (on 8 cpus) on the: (a) European RNA and (b) RDPII datasets



More dramatic improvements were observed on the RDPII dataset, as Figure 4(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. Figure 5(b) demonstrates 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.

6 Conclusions and future work

The Rec-I-DCM3 method of Roshan et al. was the first technique that allowed a successful application of parsimony heuristics with high accuracy within reasonable time limits. Nonetheless, in order to reconstruct, with high accuracy, phylogenetic trees at a much larger scale, further speed-up and improvements are imperative. In this paper we introduced the first such improvement through PRec-I-DCM3, a parallel version of the Rec-I-DCM3 method. We implemented and ran PRec-I-DCM3 on two large datasets. The results demonstrated a significant improvement over Rec-I-DCM3.

Directions for future work include

- Exploring a distributed master scheme.
- Investigating the difference in optimal parameter choice for different datasets.
- Experimental testing of PRec-I-DCM3 on simulated datasets. Using simulations allows for investigating the performance of the method with respect to the ‘true’ tree, which is known in such studies (as opposed to real datasets, in which the true tree is not known).

- Existing implementations of TNT and PAUP* are limited to handle up to 16,000-taxon trees. We intend to study the performance of PRec-I-DCM3 on datasets larger than the ones we used, once tools that handle more than 16,000 taxa are available.
- Application of Rec-I-DCM3 and PRec-I-DCM3 to Bayesian inference of phylogeny.

Acknowledgments[2]

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

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.

References

Bininda-Emonds, O.R.P. (2003) *Ratchet implementation in PAUP*4.0b10*, Available from www.tierzucht.tum.de:8080/WWW/Homepages/Bininda-Emonds.

Fitch, W.M. (1971) 'Toward defining the course of evolution: minimum change for a specified tree topology', *Syst. Zool*., Vol. 20, pp.406–416.

Foulds, L.R. and Graham, R.L. (1982) 'The Steiner problem in phylogeny is NP-complete', *Advances in Applied Mathematics*, Vol. 3, pp.43–49.

Goloboff, P.A. (1999) 'Analyzing large data sets in reasonable times: solution for composite optima', *Cladistics*, Vol. 15, pp.415–428.

Huson, D., Nettles, S. and Warnow, T. (1999a) 'Disk-covering, a fast-converging method for phylogenetic tree reconstruction', *Journal of Computational Biology*, Vol. 6, pp.369–386.

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

Maddison, D.R. (1991) 'The discovery and importance of multiple islands of most parsimonious trees', *Systematic Biology*, Vol. 42, No. 2, pp.200–210.

Maidak, B.L., Cole, J.R., Lilburn, T.G., Parker Jr., C.T., Saxman, P.R., Stredwick, J.M., Garrity, G.M., Li, B., Olsen, G.J., Pramanik, S., Schmidt, T.M. and Tiedje, J.M. (2000) 'The RDP (ribosomal database project) continues', *Nucleic Acids Research*, Vol. 28, pp.173–174.

Michener, C.D. and Sokal, R.R. (1957) 'A quantitative approach to a problem in classification', *Evolution*, Vol. 11, pp.130–162.

Nakhleh, L., Roshan, U., St. John, K., Sun, J. and Warnow, T. (2001) 'Designing fast converging phylogenetic methods', *Proc. 9th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB'01)*, Vol. 17 of *Bioinformatics*, Oxford U. Press, pp.S190–S198.

Nixon, K.C. (1999) 'The parsimony ratchet, a new method for rapid parsimony analysis', *Cladistics*, Vol. 15, pp.407–414.

Numrich, R.W. and Reid, J.K. (1998) 'Co-Array Fortran for parallel programming', *Technical Report RAL-TR-1998-060*, Rutheford Appleton Laboratory, August.

Robinson, D.F. and Foulds, L.R. (1981) 'Comparison of phylogenetic trees', *Mathematical Biosciences*, Vol. 53, pp.131–147.

Roshan, U., Moret, B.M.E., Williams, T.L. and Warnow, T. (2004a) 'Performance of supertree methods on various dataset decompositions', in Bininda-Emonds, O.R.P. (Ed.): *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, Vol. 3 of *Computational Biology*, Kluwer Academic Publishers, pp.301–328.

Roshan, U., Moret, M.E., Williams, T.L. and Warnow, T. (2004b) 'Rec-I-DCM3: a fast algorithmic technique for reconstructing large phylogenetic trees', *Proceedings of the IEEE Computational Systems Bioinformatics Conference (CSB) 2004*.

Saitou, N. and Nei, M. (1987) 'The neighbor-joining method: a new method for reconstructing phylogenetic trees', *Molecular Biology and Evolution*, Vol. 4, pp.406–425.

Sanderson, M.J., Baldwin, B.G., Bharathan, G., Campbell, C.S., Ferguson, D., Porter, J.M., von Dohlen, C., Wojciechowski, M.F. and Donoghue, M.J. (1993) 'The growth of phylogenetic information and the need for a phylogenetic database', *Systematic Biology*, Vol. 42, pp.562–568.

Swofford, D.L. (2002) *PAUP*: Phylogenetic Analysis using Parsimony (and other methods)*, Sinauer Associates, Sunderland, Mass, Version 4.0.

Warnow, T., Moret, B.M.E. and St. John, K. (2001) 'Absolute convergence: true trees from short sequences', *Proc. 12th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '01)*, SIAM Press, pp.186–195.

Williams, T.L., Moret, B.M.E., Berger-Wolf, T., Roshan, U. and Warnow, T. (2004) 'The relationship between Maximum Parsimony scores and phylogenetic tree topologies', *Technical Report TR-CS-2004-04*, Department of Computer Science, The University of New Mexico.

Wuyts, J., van de Peer, Y., Winkelmans, T. and de Wachter, R. (2002) 'The European database on small subunit ribosomal RNA', *Nucleic Acids Research*, Vol. 30, pp.183–185.

Notes

[1]The evolutionary diameter of a dataset is defined as the maximum number of changes between any two taxa in the dataset.

[2]Yuri Dotsenko and Cristian Coarfa contributed equally to this work.