DPRml: Distributed Phylogeny Reconstruction by Maximum Likelihood

Keane T.M.[1], Naughton T.J.[1*], Travers S.A.A[2], McInerney J.O.[2], McCormack, G.P. [2]

[1]Department of Computer Science, National University of Ireland, Maynooth, Ireland
[2]Department of Biology, National University of Ireland, Maynooth, Ireland
Email: tom.naughton@may.ie

Keywords: phylogeny reconstruction, phylogenetic analysis, maximum likelihood, distributed computing, Java

[*] To whom correspondence should be addressed

Abstract

Motivation: In recent years there has been increased interest in producing large and accurate phylogenetic trees using statistical approaches. However for a large number of taxa, it is not feasible to construct large and accurate trees using only a single processor. A number of specialised parallel programs have been produced in an attempt to address the huge computational requirements of maximum likelihood. We express a number of concerns about the current set of parallel phylogenetic programs which are currently severely limiting the widespread availability and use of parallel computing in maximum likelihood based phylogenetic analysis.
Results: We have identified the suitability of phylogenetic analysis to large-scale heterogeneous distributed computing. We have completed a distributed and fully cross-platform phylogenetic tree building program called DPRml. It uses an already proven maximum likelihood based tree building algorithm and a popular phylogenetic analysis library for all its likelihood calculations. It offers one of the most extensive sets of DNA substitution models currently available. We are the first, to our knowledge, to report the completion of a distributed phylogenetic tree building program that can achieve near linear speedup while only using the idle clock cycles of machines. For those in an academic or corporate environment with hundreds of idle desktop machines, we have shown how distributed computing can deliver a 'free' ML supercomputer.
Availability: The software (and user manual) is publicly available under the terms of the GNU general public licence from the system webpage at http://www.cs.may.ie/distributed
Contact: tom.naughton@may.ie

1 Introduction

One of the great challenges of molecular biology is the completion of the tree of life (Hillis and Lewis, 2000). The massive accumulation of genomic data has led to increased interest in the production of large and accurate phylogenetic trees. However the decision problem associated with searching for the best tree from a set of taxa is NP-hard (Bodlaender *et al.*, 1992). Therefore it is not feasible to perform an exhaustive search of the tree space for trees of a non-trivial size. Maximum likelihood (ML) evaluation has been widely acknowledged as one of the most accurate techniques for reconstructing phylogenies. Felsenstein first brought this framework to nucleotide-based phylogenetic inference (Felsenstein, 1981). Numerous computer studies (Huelsenbeck and Hillis, 1993; Kuhner and Felsenstein, 1994; Huelsenbeck, 1995; Rosenberg and Kumar, 2001; Ranwez and Gascuel, 2002) have shown ML programs can recover the correct tree from simulated data sets more frequently than other methods. In a recent study timing the evolution of the HIV-1 virus (Korber *et al.*, 2002), it was demonstrated that ML techniques can be effective in solving important biological problems.

Currently the most successful heuristic approach for building phylogenetic trees is to employ a hill-climbing algorithm combined with ML evaluation. Each taxon is added to the tree in a stepwise manner and topological rearrangements are subsequently performed on the best tree, in an effort to avoid local minima in the search space. The most computationally intensive aspect of this approach is that each candidate tree that is generated must have its branch lengths optimised and likelihood calculated. Some of the most popular tree building programs (Felsenstein, 1989; Rogers and Swofford, 1998) are based on this method. Despite considerable improvements in runtimes (Olsen *et al.*, 1994; Guindon and Gascuel, 2003) the single factor that is currently limiting the widespread use of ML techniques in phylogenetic analysis is the huge computational requirements (Hershkovitz and Leipe, 1998). A number of other authors (Stewart *et al.*, 2001; Stamatakis and Ludwig, 2003; Schmidt *et al.*, 2002) have concluded that the major limitation with each of these programs is that they are limited to operating on a single processor, which means that it is not feasible to build large phylogenetic trees using these programs.

In an effort to construct large and accurate phylogenetic trees while still keeping overall processing times reasonable, a number of researchers have developed parallel ML programs that utilise the stepwise insertion approach (Stewart *et al.*, 2001; Stamatakis and Ludwig, 2003). One of these programs (Stamatakis and Ludwig, 2003) also employs some simple distance based heuristics to try to reduce the number of generated trees. These programs have been successful in speeding up phylogenetic computations but the overriding problem with these programs is that specialised parallel hardware and software is often required. For most researchers, this can make these programs either prohibitively expensive or simply too complicated to set up. Furthermore these programs are often implemented in a platform specific language which imposes a restrictive limit on the numbers and types of machines that can be used in a parallel computation. It should also be noted that some of these earlier parallel programs only allowed the user to choose from a very limited number of DNA substitution models, which often leads to a poor model fit resulting in sub optimal trees. Therefore, in our opinion, the three most essential requirements of any generally usable parallel tree building program must be that the program should not require any sort of specialised or expensive parallel hardware or software, should only require the most basic technical abilities to set up and use, and should allow the user to choose from an extensive list of molecular evolution models. Currently there is no parallel phylogenetic tree building program that fulfils all of these requirements.

We have identified the suitability of phylogenetic analysis to large-scale heterogeneous distributed computing and have developed a fully cross-platform distributed application, DPRml, which we believe to be one of the most general and powerful likelihood-based phylogenetic tree building programs currently available. DPRml is, to our knowledge, the first distributed phylogenetic tree building program to satisfy each of the three requirements outlined above. The generality of our program is demonstrated by the fact that DPRml, written in Java, can run on virtually any architecture and operating system simultaneously while only using the spare clock cycles of donor machines. No specialised computer hardware is required, and no expense is incurred if idle computing resources are harnessed. This would not be as straightforward for a distributed application written in a native language because the application would have to be compiled for each particular architecture and operating system. We have demonstrated the ease of use and platform heterogeneity of DPRml with experiments that utilise the spare computing resources of several different architectures and operating systems simultaneously. The user has a very straightforward configuration file with which to tailor the computation and can choose from one of the most extensive ranges of DNA substitution models currently available. Our performance analysis demonstrates how effective DPRml can be for speeding up the process of constructing large phylogenetic trees. DPRml implements an already proven tree building algorithm (Stewart *et al.*, 2001; Olsen *et al.*, 1994) and uses the popular Phylogenetic Analysis Library (PAL) v1.4 (Drummond and Strimmer, 2001) for all its likelihood calculations.

DPRml is just one application of large-scale distributed computing. Our intention with this paper is to highlight the general applicability of this computing paradigm to certain bioinformatics computations. By a detailed presentation of this specific example we wish to highlight the hallmarks of the paradigm which are ease-of-use, flexibility, affordability, and efficiency.

2 Distributed Computing and Phylogenetic Analysis

In recent years, the area of distributed computing has emerged as a viable alternative to specialised parallel computing. By harnessing the spare clock cycles of idle machines (Buyya, 1999), it is possible to emulate the computing power offered by a specialised parallel machine at a fraction of the cost. Several successful systems have been developed on this basis, e.g. Seti@Home (Korpela *et al.*, 2001), Folding@Home (Larson *et al.*, 2003), Condor (Thain *et al.*, 2003), and Models@Home (Kreiger and Vriend, 2002). The type of applications that are generally considered to be suited to distributed computing have the capability to fully exploit "coarse-grained parallelism," meaning that it should be possible to partition the application into independent tasks or processes that can be computed concurrently. Typically these types of problems must display a high "compute-to-data" ratio to make it worthwhile sending the data over a network rather than computing locally.

The process of constructing large phylogenetic trees using ML analysis generates thousands of candidate trees that must have their branch lengths optimised and likelihood calculated. These two processes can be done completely independently for each tree and the set of trees generated at each stage can be represented, stored, and transferred compactly using only a few KBytes. Therefore the small size of the data involved coupled with the long computation times of ML analysis make this problem ideal for a large scale distributed computing implementation.

2.1 DPRml Algorithm

We have taken the hill-climbing algorithm used by parallel fastDNAml (Stewart *et al.*, 2001) and have implemented a platform independent, distributed, and much more generalised version of the program. The algorithm implemented by DPRml is outlined in Fig. 1. The parameters *m* and *v* are contained in the parameter file. Step 1 of the algorithm is a localised version of the overall algorithm outlined in steps 2-7. A single donor machine builds an initial tree for *m* minutes (default value is 30 minutes) as it is more efficient to build this initial tree on a single donor machine than to distribute this part of the computation. At each stage of the algorithm, the set of generated trees is split into equal size groups and issued to clients on a first come, first served basis. The inputs to the application are a MODELTEST (Posada and Crandall, 1998) output file, a FASTA sequence file (DNA or RNA), and an input parameter file. The outputs of the program are a Nexus format tree file, a Newick format tree file, a PAL tree object file, a human readable tree (text file), and the likelihood of the final tree. We have provided a remote interface to the system that makes it possible to monitor the progress of the application in real-time as it builds the phylogenetic tree. The program supports all of the DNA substitution models that MODELTEST v3.06 provides. The input parameter file lets the user set various runtime options for the computation such as the maximum number of vertices that rearrangements can span, whether to keep a copy of the best tree from every stage or just the best tree from the previous stage, whether to add the taxa in a randomly generated order or input order, and whether to optimise the branch lengths of every tree that is generated or just optimise the final tree. If there was a catastrophic event (e.g. a system wide power failure), it is possible for the program to continue building the tree from where it left off. Log files make it possible to fully examine and track the entire tree building process.

2.2 Implementation

DPRml is implemented entirely in Java, meaning that the program is completely platform and network independent. DPRml is just one of the applications that can run on our general purpose distributed computing platform, loosely based on the design of the Java Distributed Computing Library (JDCL) (Fritsche *et al.*, 2001; Keane *et al.,* 2003). In our deployment of

DPRml, we have our client software running in a number of computing laboratories, consisting of approximately 200 desktop PC's of various modest specifications (Pentium II's up to Pentium IV's running assorted versions of Windows and Linux OSs). To minimise disruption to users, we run the client as a low priority background service that only uses the idle clock cycles of the machines. To illustrate the portability of our system, we have also installed our client on every node of an IBM Linux cluster (32 Dual PIII 1 GHz nodes with between 256-768 MB memory per node) with the desktops and cluster nodes connecting to a single server.

3 Performance Analysis

A number of standard measures have emerged in parallel computing for measuring the performance of parallel programs. Running time measures the amount of time from when a parallel program is started to when the program produces the final result of the computation. Speedup *s* is the ratio between the running times using one processor and multiple processors. It measures the performance improvement gained through parallelisation and is calculated from

$$s(n) = t(1) / t(n)\,, \tag{1}$$

where *t(1)* is the running time of the program using a single processor, and *t(n)* is the running time of the program using *n* processors. The maximum theoretical speedup occurs when there is an *n* times speedup achieved using *n* processors. The ideal speedup curve is rarely achieved because parallelism entails a certain amount of communication and management overhead. It should also be noted that the maximum speedup achievable depends greatly on the degree of parallelism in a particular algorithm (Amdahl, 1967). Scalability is the ability to maintain performance levels as the workload increases by incrementally adding more system capacity (adding more processors and/or computations running simultaneously).

We performed a full set of performance tests using the dataset that was used to benchmark parallel fastDNAml (Stewart *et al.*, 2001). This dataset consists of three individual sets of taxa consisting of 50 taxa, 101 taxa, and 150 taxa that are 1858 (50 and 101 taxa) and 1269 (150 taxa) nucleotide positions in length. In our tests, we used the HKY (Hasegawa *et al.*, 1985) DNA substitution model with the same three Ts/Tv ratio parameters as were used by parallel fastDNAml. We examined several trees constructed by DPRml using this dataset and found that there were only minor differences due to the differing randomisation of the taxa addition order. Several of the trees produced are available from the system webpage. For all of our performance tests, we ran a version of the program that adds the taxa to the tree in the same order each time (so that the scaling behaviour of the program could be clearly understood) and the program was configured to optimise the branch lengths of every tree generated. The maximum number of vertices that rearrangements could cross was set to five.

We compared the single-processor performance of DPRml and fastDNAml (Olsen *et al.*, 1994) using the three datasets. The results of these tests are shown in Table 1. Although DPRml performs on average 7 times slower than fastDNAml, DPRml's performance reduction is overcome its greater cross-platform compatibility.

3.1 Single Problem Speedup Analysis

To analyse the speedup that can be gained by running DPRml, we ran a single instance of DPRml on the distributed system with differing numbers of clients and noted the total running time in each case. For these particular tests, the set of clients consisted of two university computing laboratories with a total of 60 desktop PC's (each machine was a Pentium IV 2.4

GHz with 512 Mbytes of memory running either Windows 2000 or Redhat Linux 7.0). Our server resided on a Pentium III 600 MHz with 256 Mbytes of memory running Debian Linux with a 10 Mbit/s connection to the laboratories. We had our client installed as a low priority background service and the PC's were in use, and were being rebooted between operating systems during teaching hours. The graphs show the corresponding mean running time decrease (Fig. 2) and speedup gained (Fig. 3) over two runs for each point on the graphs. The main factor limiting the scalability of the program is the synchronisation barrier created by the staged nature of the algorithm. If any of the donor machines are unexpectedly switched off, DPRml must wait for the distributed system to detect this and redistribute the data to another donor machine before it can proceed to the next stage of the algorithm. Figure 3 shows that DPRml scales extremely well, with the speedup increasing with an increase in dataset size. This is consistent with the findings of a special purpose parallel phylogenetic program (Stewart *et al.*, 2001). For this particular dataset, it is expected that the speedup gains should plateau at approximately 150 processors because at this point the number of processors would equal the number of trees being generated at many of the stages. This would also be the case for any other parallel tree building program.

3.2 Multiple Problem Efficiency Analysis

One way to maintain consistently high efficiency (utilisation of donor machines) in the distributed system is to run several DPRml computations simultaneously. Ideally, each computation would be at a different stage in the tree building algorithm and therefore should result in consistently higher overall efficiency. Multiple DPRml computations can be submitted to the server, which allows users to always make optimal use of the available donor machines. We wanted to investigate fully how to optimise the efficiency of the distributed system by running differing numbers of DPRml computations simultaneously. We were also interested in the extent to which an increasing number of DPRml computations running simultaneously would affect the rate at which the phylogenetic trees are built. To investigate these two related issues, we used one of the datasets that was used to test parallel fastDNAml (Stewart *et al.*, 2001), consisting of 101 taxa (1858 nucleotides per taxa), and we ran varying numbers of DPRml computations on the system while keeping the number of donor machines fixed. The set of clients consisted of a university computing laboratory with a total of 40 desktop PC's (each machine was a Pentium III 600 MHz with 128 Mbytes of memory running Windows NT). By examining the distributed system log files, we completed a graph (see Fig. 4) showing the efficiency of the system over a period of 24 hours for each set of problems. We also noted the average size of the trees built (see Table 2) at the end of each 24 hour period.

Figure 4 shows that efficiency is greatly increased when the number of tree building computations running simultaneously in the system is increased. Table 2 is quite interesting as it shows that by increasing the number of tree building computations from one to six only reduced the average tree size by 31%. For this particular dataset and set of donor machines, six tree building computations is sufficient to get almost 100% efficiency from the system. To further investigate the effect on speedup of running multiple DPRml computations in the distributed system, we completed a speedup graph (see Fig. 5) based on the running time of six simultaneous DPRml computations. For this test, we used one of the datasets that was used to test parallel fastDNAml (Stewart *et al.*, 2001), consisting of 50 taxa (1858 nucleotides per taxa), and ran six simultaneous computations with varying numbers of clients. As expected, Fig. 5 demonstrates that DPRml achieves near linear speedup when speedup is measured with multiple DPRml computations running simultaneously.

The above results fit well with the expected usage of the program. As the algorithm outlined in section 2 is heuristic, it is possible to become trapped in a local optimum, rather than a global one. Typically a researcher would repeat the entire tree building process with several different randomisations of the taxa addition order and then compare the best of the resulting trees to determine a consensus tree (Jermiin *et al.*, 1997). As has been noted by parallel computing authors (Amdahl, 1967), it is quite rare and difficult for a parallel or distributed system to achieve 100% efficiency. We are the first, to our knowledge, to report the completion of a distributed phylogenetic tree building program that can achieve near linear speedup and almost 100% system efficiency while only using the idle clock cycles of standard desktop machines.

4 Discussion

DPRml is an easy-to-use practical application that can harness the idle computing resources of any research institute to construct large phylogenetic trees using ML. The real significance of DPRml lies in the fact that it gives a researcher, who may not have access to (or the technical skills necessary to access) a dedicated parallel machine, the ability to build large and accurate phylogenetic trees. Unlike other parallel phylogenetic programs, no specialist parallel computing knowledge is required to set up and run DPRml. The program offers an extensive list of DNA substitution models that allows users to pick the substitution model that better reflect their dataset. We have shown how effective DPRml can be for speeding up phylogenetic computations by performing a full performance analysis. The final outputs of the program are in standard formats that allow the user to perform further manipulation and analysis of results using other phylogenetic packages.

This first release of DPRml uses PAL v1.4 (Drummond and Strimmer, 2001) for all of its optimisation and likelihood calculations. As new features and algorithmic improvements appear in later versions of PAL (Goode *et al.*, 2004), we will release updated versions of DPRml on our webpage to take advantage of the improvements. In future versions of DPRml we plan to focus our investigations on algorithmic improvements and plan to add features such as bootstrap analysis and supertree construction. On the wider issue of the large-scale distributed computing paradigm, we have highlighted the principal advantages of the paradigm, which are ease-of-use, flexibility, affordability, and efficiency.

Acknowledgements

This research has been funded by Embark Initiative from the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan. We would also like to thank Matthew Goode of the PAL project for help and advice on how to use PAL v1.4.

References

Amdahl, G.M. (1967) Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities, In *AFIPS Conference Proceedings*, 30, 483-485, AFIPS Press, Reston, Va

Bodlaender, H., Fellows, M., and Warnow, T. (1992) Two strikes against perfect phylogeny, *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming,* Lecture Notes in Computer Science, 623, 273-283, Springer-Verlag, NY

Bull, J.M, Smith, L.A., Pottage, L., and Freeman, R. (2001) Benchmarking Java against C and Fortran for Scientific Applications, *ACM 2001 Java Grande/ISCOPE Conference*, 97-105, San Francisco, CA

Buyya, R., editor (1999) High Performance Cluster computing: Architectures and Systems, Prentice Hall Inc., ISBN-0130-1378-47

Drummond, A. and Strimmer, K. (2001) PAL: An object-oriented programming library for molecular evolution and phylogenetics, *Bioinformatics*, 17, 662-663

Felsenstein, J. (1989) PHYLIP -- Phylogeny Inference Package (Version 3.2), *Cladistics*, 5, 164-166

Felsenstein, J. (1981) Evolutionary trees from DNA sequences: A maximum likelihood approach, *Journal of Molecular Evolution*, 17, 368-376

Fritsche, K., Power, J., and Waldron, J. (2001) A Java distributed computation library, *Proceedings of the 2nd International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT2001)*, 236-243, Taipei, Taiwan

Goode, M., Strimmer, K., Drummond, A., Buckler, E., and Rodrigo A. (2004) A brief introduction to the phylogenetic analysis library, version 1.5, *Proceedings of the Second Asia-Pacific Bioinformatics Conference (APBC2004),* Dunedin, NZ

Guindon, S. and Gascuel, O. (2003) A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood, *Systematic Biology*, 52(5), 696-704

Hasegawa, M., Kishino, H., and Yano, T. (1985) Dating the human-age splitting by a molecular clock of mitochondrial DNA, *Journal of Molecular Evolution*, 22, 160-174

Hershkovitz, M.A., and Leipe, D.D. (1998) Bioinformatics: a practical guide to the analysis of genes and proteins, A.D. Baxevanis and B.F.F. Ouelette (Eds.), 189-230, Wiley-Liss, New York

Hillis, D.M., and Lewis, P. (2000) Computing the Tree of Life, *Envision*, 16 (3)

Huelsenbeck, J.P. (1995) Performance of phylogenetic methods in simulation, *Systematic Biology*, 44, 17–48

Huelsenbeck, J.P., and Hillis, D.M. (1993) Success of phylogenetic methods in the four-taxon case, *Systematic Biology*, 42, 247–264

Jermiin, L. S., Olsen, G.J., and Easteal, S. (1997) Majority rule consensus of maximum likelihood trees, *Molecular Biology and Evolution*, 14, 1296–1302

Keane, T., Allen, R., Naughton, T., McInerney, J., and Waldron, J. (2003) Distributed Java platform with programmable MIMD capabilities, in Guelfi, N., Astesiano, E., Reggio, G. (Eds.), *Scientific Engineering for Distributed Java Applications*, Lecture Notes in Computer Science, 2604, 122-132, Springer, Berlin

Korber, B., Muldoon, M., Theiler, J., Gao, F., Gupta, R., Lapedes, A., Hahn, B.H., Wolinsky, S., and Bhattacharya, T. (2000) Timing the ancestor of the HIV-1 pandemic strains, *Science*, 288, 1789-1796

Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Lebofsky, M. (2001) SETI@home-Massively Distributed Computing for SETI, *IEEE: Computer Science and Engineering*, 3 (1), 77-83

Kuhner, M.K., and Felsenstein, J. (1994) A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates, *Molecular Biology and Evolution*, 11, 459–468

Larson, S.M., Snow, C.D., Shirts, M.R., Pande. V.S., (2003) Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology, to appear in *Computational Genomics*, Richard Grant, editor, Horizon Press

Olsen, G.J., Matsuda, H., Hagstrom, R., and Overbeek, R. (1994) FastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood, *Computer Applications in the Biosciences*, 10, 41-48

Posada, D. and Crandall, K.A. (1998) MODELTEST: testing the model of DNA substitution, *Bioinformatics* 14 (9), 817-818

Ranwez, V., and Gascuel, O. (2002) Improvement of distance-based phylogenetic methods by a local maximum likelihood approach using triplets, *Molecular Biology and Evolution*, 19, 1952–1963

Rogers, J. S. and Swofford D. L. (1998) A fast method for approximating maximum likelihoods of phylogenetic trees from nucleotide sequences, *Systematic Biology*, 47, 77-89

Rosenberg, M., and Kumar, S. (2001) Traditional phylogenetic reconstruction methods reconstruct shallow and deep evolutionary relationship equally well, *Molecular Biology and Evolution*, 19, 1823–1827

Schmidt, H.A., Strimmer, K., Vingron, M., and von Haeseler, A. (2002) TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing, *Bioinformatics*, 18, 502-504

Stamatakis, A.P., and Ludwig, T. (2003) Phylogenetic Tree Inference on PC Architectures with AxML/PAxML, *Proceedings of IPDPS2003 (High Performance Computational Biology workshop)*, 157, Nice, France

Stewart, C.A., Hart, D., Berry, D.K., Olsen, G.J., Wernert, E.A., and Fischer, W. (2001) Parallel implementation and performance of fastDNAml – a program for maximum likelihood phylogenetic inference, *Proceedings of SC2001*, Denver, CO, USA

Thain, D., Tannenbaum, T., and Livny, M. (2003) Condor and the Grid, in Berman, F., Hey, A., and Fox, G. editors, *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley

Figure Captions

Figure 1. Tree building algorithm implemented by DPRml

Figure 2. Decrease in computation time with an increase in the number of processors over each of the three datasets (50, 101, and 150 taxa). The average over two runs for each dataset is shown

Figure 3. Speedup achieved over each of the three datasets (50, 101, and 150 taxa). The average over two runs for each dataset is shown. Linear speedup is the theoretical maximum for parallel algorithms

Figure 4. Efficiency of the system over a period of 24 hours for varying numbers of DPRml computations running in the system

Figure 5. Speedup achieved over 50 taxa dataset with 6 problems running simultaneously. Linear speedup is the theoretical maximum for parallel algorithms

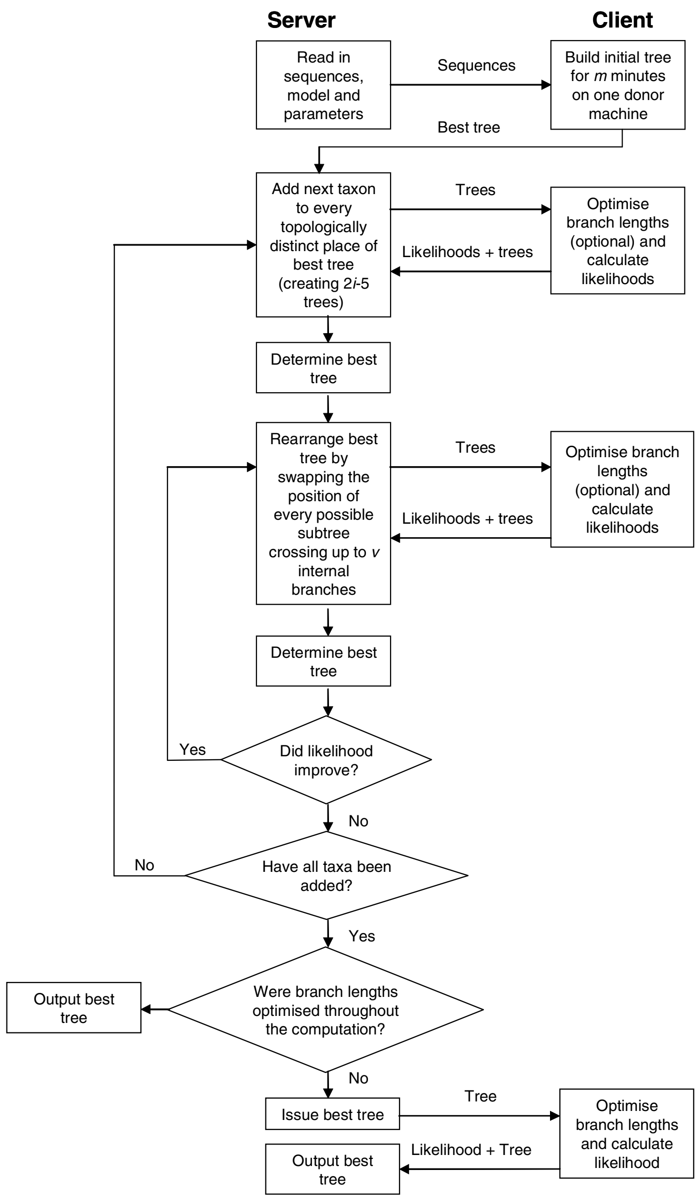
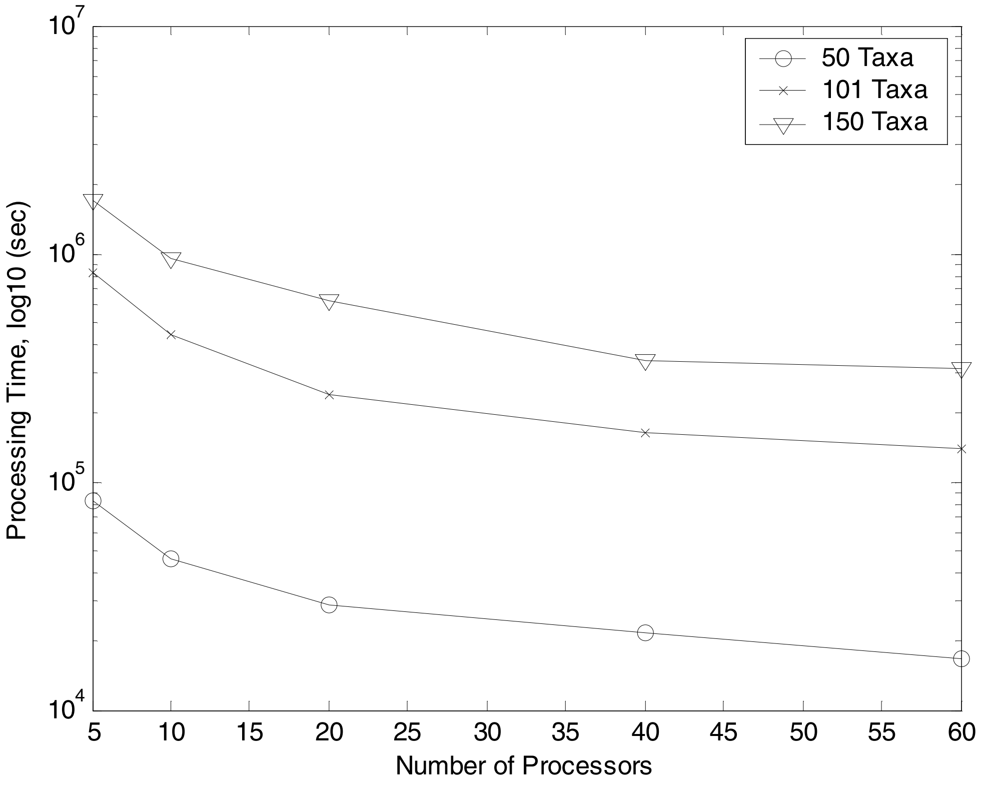
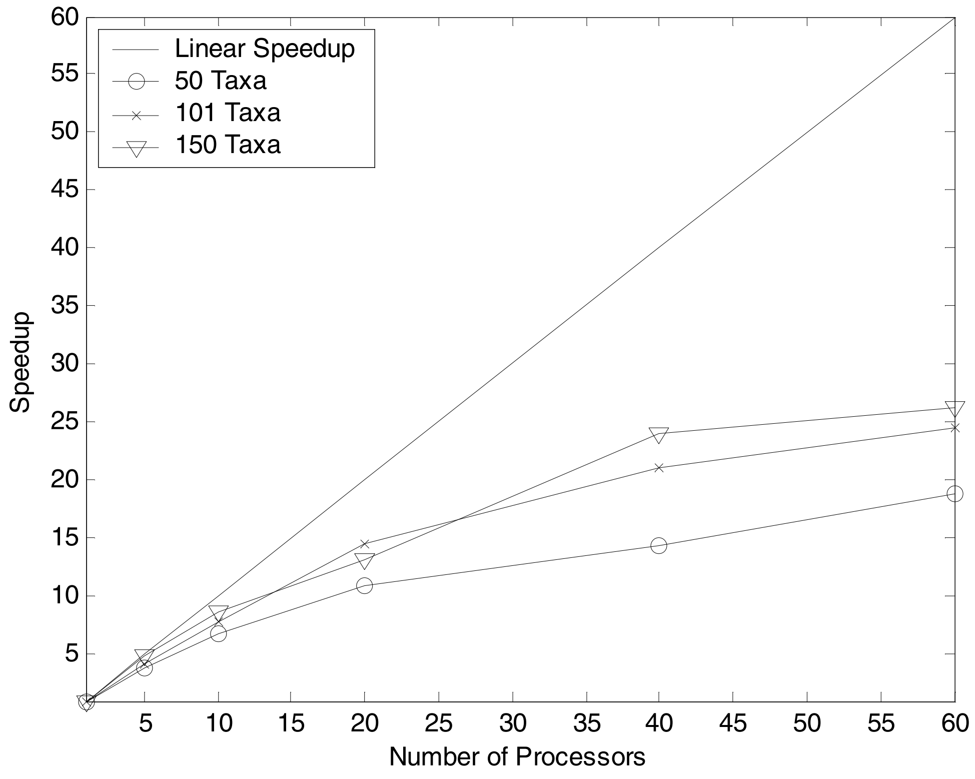

Server
Client
Read in sequences, model and parameters
Sequences
Build initial tree for *m* minutes on one donor machine
Best tree
Add next taxon to every topologically distinct place of best tree (creating 2*i*-5 trees)
Trees
Likelihoods + trees
Optimise branch lengths (optional) and calculate likelihoods
Determine best tree
Rearrange best tree by swapping the position of every possible subtree crossing up to *v* internal branches
Trees
Likelihoods + trees
Optimise branch lengths (optional) and calculate likelihoods
Determine best tree
Did likelihood improve?
Yes
No
Have all taxa been added?
No
Yes
Were branch lengths optimised throughout the computation?
Output best tree
No
Issue best tree
Tree
Optimise branch lengths and calculate likelihood
Likelihood + Tree
Output best tree

Fig. 1



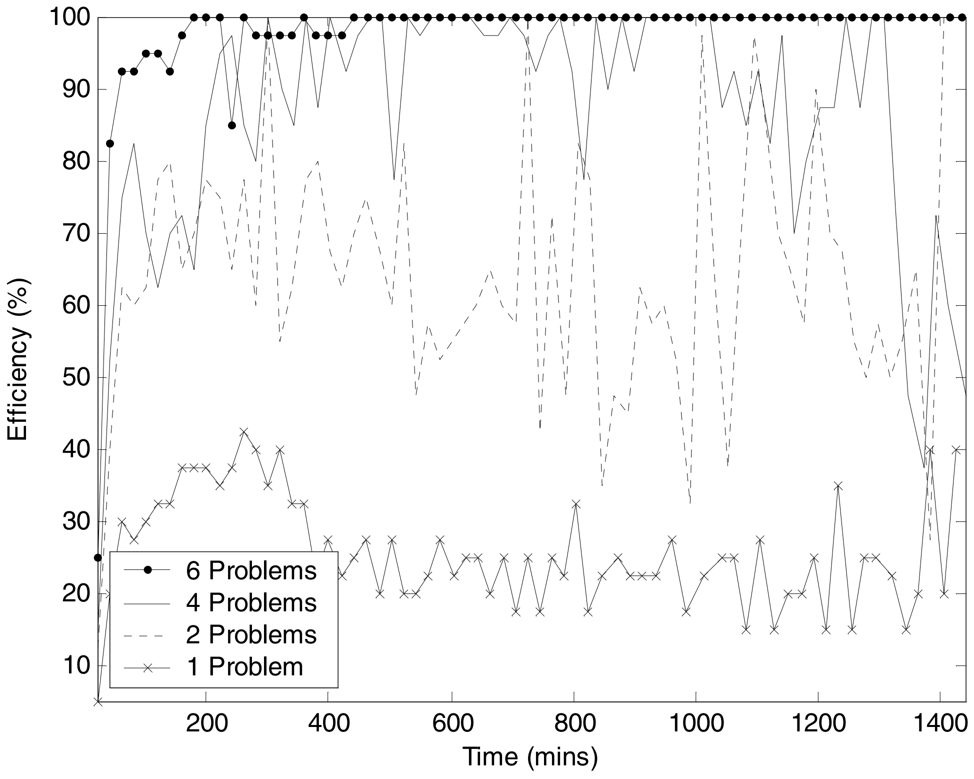
50 Taxa
101 Taxa
150 Taxa
Processing Time, log10 (sec)
Number of Processors
10^4
10^5
10^6
10^7
5
10
15
20
25
30
35
40
45
50
55
60

Fig. 2



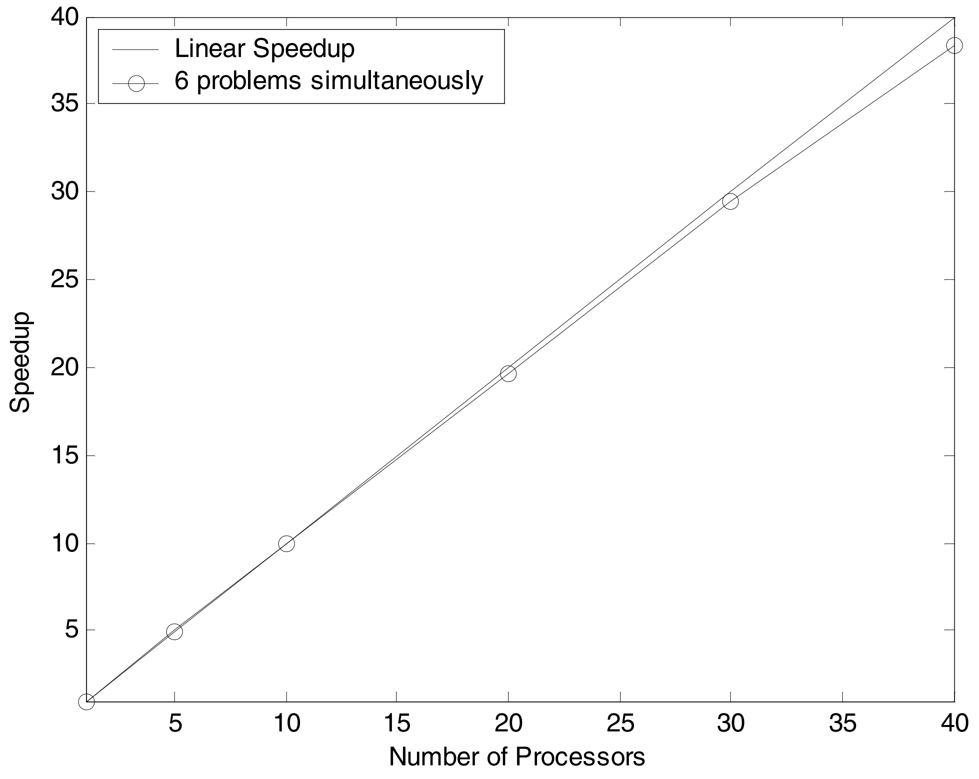
Linear Speedup
50 Taxa
101 Taxa
150 Taxa
Speedup
Number of Processors
5
10
15
20
25
30
35
40
45
50
55
60

Fig. 3



Efficiency (%)
100
90
80
70
60
50
40
30
20
10
200
400
600
800
1000
1200
1400
Time (mins)
6 Problems
4 Problems
2 Problems
1 Problem

Fig. 4



Linear Speedup
6 problems simultaneously
Speedup
Number of Processors
40
35
30
25
20
15
10
5
5
10
15
20
25
30
35
40

Fig. 5

	50 Taxa	101 Taxa	150 Taxa
DPRml	1386	57373	123484
FastDNAml	240	8726	14685

Table 1. Runtime comparison of DPRml and FastDNAml for the three datasets (50, 101 and 150 taxa). All times are in minutes.

# Problems	1	2	4	6
Tree Size	72	58	54	49

Table 2. Average tree size after 24 hours for varying numbers of DPRml problems running simultaneously in the distributed system