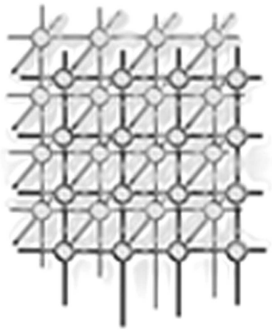
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2003; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

RAxML-II: A Program for Sequential, Parallel & Distributed Inference of Large Phylogenetic Trees



Alexandros Stamatakis[1,*,*] , Thomas Ludwig[2], Harald Meier[1]

[1] *Technische Universität München, Lehrstuhl für Rechnertechnik und Rechnerorganisation/I10, Boltzmannstr. 3 D-85748 Garching b. München, Germany*
[2] *Ruprecht-Karls-Universität Heidelberg, Institut für Informatik, Im Neuenheimer Feld 348, D-69120 Heidelberg, Germany*

SUMMARY

Inference of phylogenetic trees comprising hundreds or even thousands of organisms based on the maximum likelihood method is computationally intensive. We present simple heuristics which yield accurate trees for synthetic as well as real data and significantly reduce execution time. Those heuristics have been implemented in a sequential, parallel, and distributed program called RAxML-II which is freely available as open source code. We compare performance of the sequential program with PHYML and MrBayes which -to the best of our knowledge- are currently the fastest and most accurate programs for phylogenetic tree inference based on statistical methods. Experiments are conducted using 50 synthetic 100 taxon alignments as well as 9 real-world alignments comprising 101 up to 1.000 sequences. RAxML-II outperforms MrBayes for real-world data both in terms of speed and final likelihood values. Furthermore, for real data RAxML-II requires less time (factor 2–8) than PHYML to reach PHYML's final likelihood values and yields better final trees due to its more exhaustive search strategy. For synthetic data MrBayes is slightly more accurate than RAxML-II and PHYML but significantly slower. The non-deterministic parallel program shows good speedup values and has been used to infer a 10.000-taxon tree comprising organisms from the domains: Eukarya, Bacteria, and Archaea

KEY WORDS: Phylogenetic trees; maximum likelihood; parallel and distributed computing

*Correspondence to: Technische Universität München, Lehrstuhl für Rechnertechnik und Rechnerorganisation/I10, Boltzmannstr. 3 D-85748 Garching b. München, Germany
*E-mail: stamatak@cs.tum.edu
Contract/grant sponsor: This work is sponsored under the project ID **ParBaum**, within the framework of the "Competence Network for Technical, Scientific High Performance Computing in Bavaria": Kompetenznetzwerk für Technisch-Wissenschaftliches Hoch- und Höchstleistungsrechnen in Bayern (KONWIHR). KONWIHR is funded by means of "High-Tech-Offensive Bayern". All major parallel tests have been carried out on the Linux Cluster System of the Regionales RechenZentrum Erlangen (RRZE).

Received 31 January 2000
Revised 19 September 2002

1. INTRODUCTION

Within the ParBaum (Parallel Tree) project at the Technical University of Munich (TUM), work is conducted on phylogenetic tree inference based on the maximum likelihood method by J. Felsenstein [2]. The overall aim of the project is to develop novel systems and algorithms for the computation of huge phylogenetic trees based on sequence data from the ARB [13] small subunit ribosomal RiboNucleic Acid (ssu rRNA) database in distributed and parallel environments. In previous work [25] we have introduced Subtree Equality Vectors (SEVs) as a means to significantly reduce topology evaluation time. Topology evaluation represents the by far most cost-intensive part of every phylogenetic tree inference process based on the maximum likelihood method irrespective of the tree building algorithm deployed. We implemented the SEV-concept in parallel fastDNAml [14, 26] and named the resulting program PAxML (Parallel Axelerated Maximum Likelihood). In tests with alignments of 150 up to 500 sequences, we achieved global run time improvements of 26% up to 65% compared to parallel fastDNAml.

In this paper we present simple new heuristics which significantly accelerate the tree optimization process and yield accurate results. The heuristics have been implemented in a sequential program called RAxML-II (Randomized Axelerated Maximum Likelihood). In addition, we present a parallel and distributed seti@home-like implementation of RAxML-II. The parallel and distributed versions allow for inference of huge trees on inexpensive hardware architectures.

The remainder of this paper is organized as follows: In Section 1.1 we briefly describe related work with a focus on current state-of-the-art programs for maximum likelihood-based and bayesian phylogenetic inference, which we deploy to assess performance of RAxML-II. In the following Sections 2.1, 2.2, and 2.3 we describe the new heuristics as well as the design of the parallel and distributed algorithm. Experimental results for the sequential and parallel program are summarized in Section 3. We conclude in Section 4 and briefly address current and future issues of research.

1.1. Related work

We limit our survey of related work to statistical methods since they have shown to be the most accurate methods currently available. On the one hand there exist "traditional" maximum likelihood methods and a large variety of programs implementing maximum likelihood searches. The site [19] maintained by J. Felsenstein lists most available programs. On the other hand bayesian methods have emerged which are relatively new compared to maximum likelihood and have experienced great impact, especially through the release of a program called MrBayes [8].

A thorough comparison of popular phylogeny programs using statistical approaches such as fastDNAml, MrBayes, PAUP [18], and treepuzzle [27] based on synthetic data has been conducted by T.L. Williams et al. [30]. The most important result of this paper is that MrBayes outperforms all other phylogeny programs in terms of speed and tree quality.

MrBayes carries out bayesian inference of phylogenetic trees using the Metropolis-Coupled Markov Chain Monte Carlo (MC^3) technique.

Recently, Guidon and Gascuel published an interesting paper about their new program PHYML [4], which is very fast and seems to be able to compete with MrBayes. PHYML is a "traditional" maximum likelihood program which seeks to find the tree topology which maximizes the likelihood value. Like MrBayes, PHYML is also capable to optimize evolutionary model parameters.

Thus, -to the best of our knowledge- MrBayes and PHYML are currently the fastest and most accurate representatives of bayesian and maximum likelihood approaches to phylogenetic tree inference. Therefore, we focus on those two programs for assessing performance of RAxML-II. A plethora of genetic maximum likelihood search algorithms has been devised as well. For example the program MetaPIGA [12] represents a very efficient implementation of a genetic algorithm for maximum likelihood-based tree inference. However, genetic algorithms are generally slower than PHYML or RAxML-II.

An important point regarding performance analysis of these programs is that one should be careful when comparing bayesian with maximum likelihood methods due to subtle differences in the statistical models. This is due to the fact that bayesian methods optimize the integrated likelihood values over a broad range of topologies and model parameters, whereas maximum likelihood methods seek to find the topology with the peak likelihood value. Thus, a bayesian analysis might not yield the peak likelihood values as obtained from a maximum likelihood search. A useful comparison of traditional and bayesian phylogenetic inference methods can be found in [7].

In what concerns parallel computing, the parallel implementations of bayesian methods are relatively closely coupled such that high performance computers with expensive communication infrastructure are required [3]. In fact, similar parallelization techniques as for numerical simulations are deployed. For PHYML there exists no parallel implementation yet. The parallel implementation of fastDNAml [26] is still widely used, despite the fact that it is based on the old sequential fastDNAml algorithm dating from 1994. Finally, genetic search algorithms have also been parallelized to compute relatively small trees comprising up to 228 sequences [1].

2. NEW HEURISTICS

2.1. Sequential algorithm

The heuristics of RAxML-II belong to the class of algorithms, that optimize the likelihood of a starting tree which already comprises all sequences. In contrast to other programs RAxML-II starts by building an initial parsimony tree with dnapars from Felsenstein's PHYLIP package [19] for two reasons:

Firstly, parsimony is related to maximum likelihood under simple evolutionary models [29], such that one can expect to obtain a starting tree with a relatively good likelihood value compared to random or neighbor joining starting trees. For example, the 500_ZILLA parsimony starting tree showed a better likelihood than the final tree of PHYML (see Table I).

Secondly, dnapars uses stepwise addition [2] for tree building and is relatively fast. The stepwise addition algorithm enables the construction of distinct starting trees by using a randomized input sequence order. Thus, RAxML-II can be executed several times with different starting trees and thereby compute a set of distinct final trees. The set of final trees can be used to build a consensus tree and augment confidence into the final result since RAxML-II explores the search space from different starting points. To speed up computations, some optimization steps have been removed from dnapars.

The tree optimization process represents the second and most important part of the heuristics. RAxML-II performs standard subtree rearrangements by subsequently removing all possible subtrees from the currently best tree t_{best} and re-inserting them into neighboring branches up to a specified distance of nodes. RAxML-II inherited this optimization strategy from fastDNAml. One rearrangement step in fastDNAml consists of moving all subtrees within the currently best tree by the minimum up to the maximum distance of nodes specified (lower/upper rearrangement setting). This process is outlined for a single subtree (ST5) and a distance of 1 in Figure 1 and for a distance of 2 in Figure 2 (not all possible moves are shown). In fastDNAml the likelihood of each thereby generated topology is evaluated by exhaustive branch length optimizations. If one of those alternative topologies improves the likelihood t_{best} is updated accordingly and once again all possible subtrees are rearranged within t_{best}. This process of rearrangement steps is repeated until no better topology is found.

The rearrangement process of RAxML-II differs in two major points: In fastDNAml after each insertion of a subtree into an alternative branch the branch lengths of the entire tree are optimized. As depicted in Figures 1 and 2 with bold lines RAxML-II only optimizes the three local branches adjacent to the insertion point of the subtree either analytically (fast) or by the Newton-Raphson method (slower) before computing its likelihood value. Since the likelihood of the tree strongly depends on the topology per se this fast pre-scoring can be used to establish a small list of potential alternative trees which are very likely to improve the score of t_{best}. RAxML-II uses a list of size 20 to store the best 20 trees obtained during one rearrangement step. This list size proves to be a practical value in terms of speed and thoroughness of the search. After completion of one rearrangement step the algorithm performs global branch length optimizations on those 20 best topologies only. The capability to analyze significantly more alternative and diverse topologies due to a computationally feasible higher rearrangement setting (e.g. 5 or 10) leads to significantly improved final trees.

Another important change especially for the initial optimization phase, i.e. the first 3-4 rearrangement steps, consists in the subsequent application of topological improvements during one rearrangement step. If during the insertion of one specific subtree into an alternative branch a topology with a better likelihood is encountered this tree is kept immediately and all subsequent subtree rearrangements of the current step are performed on the improved topology. The mechanism is outlined in Figure 3 for a subsequent application of topological improvements via subtree rearrangements of ST5 and ST3 on the same initial tree. This enables rapid initial optimization of random starting trees as depicted e.g. for two alignments containing 150 taxa in Figures 9 and 10.

The exact implementation of the RAxML-II algorithm is indicated in the C-like pseudocode below. The algorithm is passed the user/parsimony starting tree `t`, the initial rearrangement setting `rStart` (default: 5) and the maximum rearrangement setting `rMax` (default: 21).

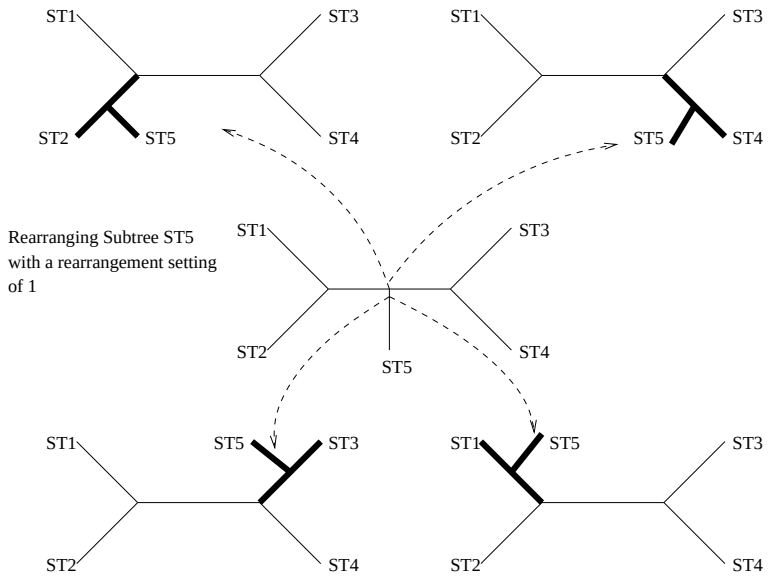


Figure 1. Rearrangements traversing one node for subtree ST5, branches which are optimized are indicated by bold lines

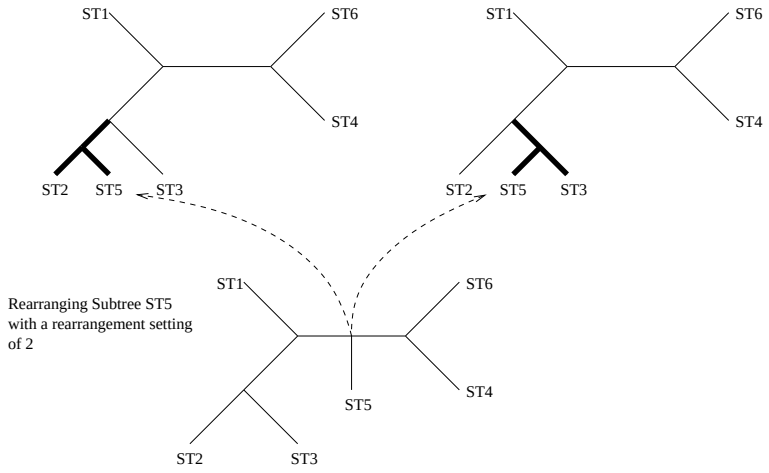


Figure 2. Example rearrangements traversing two nodes for subtree ST5, branches which are optimized are indicated by bold lines

Initially, the rearrangement stepwidth ranges from `rL = 1` to `rU = rStart`. Fast analytical local branch length optimization `a` is turned off when functions `rearr(...)`, which actually performs the rearrangements, and `optimizeList20()` fail to yield an improved tree for the first time. As long as the tree does not improve the lower and upper rearrangement parameters `rL, rU` are incremented by `rStart`. The program terminates when the upper rearrangement setting is greater or equal to the maximum rearrangement setting, i.e. `rU >= rMax`.

```
RAxML-II(tree t, int rStart, int rMax)
{
 int rL, rU;
 boolean a = TRUE;
 boolean impr = TRUE;

 while(TRUE)
  {
   if(impr)
    {
     rL = 1;
     rU = rStart;
     rearr(t, rL, rU, a);
    }
   else
    {
     if(!a)
      {
       a = FALSE;
       rL = 1;
       rU = rStart;
      }
     else
      {
       rL += rStart;
       rU += rStart;
      }
     if(rU < rMax)
      rearr(t, rL, rU, a);
     else
      goto end;
    }
   impr = optimizeList20();
  }
 end:
}
```

2.2. Parallel algorithm

The parallel implementation is based on a simple master-worker architecture and consists of two phases. In **phase I** the master distributes the alignment file to all worker processes if no common file system is available, otherwise it is read directly from the file. Thereafter, each worker independently computes a randomized parsimony starting tree and sends it to the master process. Alternatively, it is possible to start the program directly in **phase II** by specifying a tree file name in the command line. In **phase II** the master initiates the optimization process for the best parsimony or specified starting tree. Due to the high

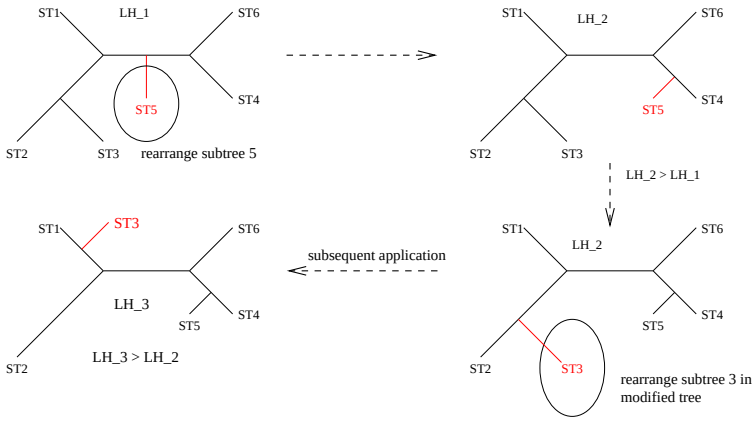


Figure 3. Example for subsequent application of topological improvements during one rearrangement step

speed of a single topology evaluation as well as the requirement for atomicity of a specific subtree rearrangement by function `rearrangeSubtree()` and the high communication cost, it is not feasible to distribute work by single topologies as e.g. in parallel fastDNAml. Therefore, we distribute work by sending the subtree ID (of the subtree to be rearranged) along with the currently best topology `t_best`, to each worker. The `sequential` and `parallel` implementation of RAxML-II on the master-side is outlined in the pseudocode of function `rearr()` which actually executes subtree rearrangements. Each worker simply executes function `rearrangeSubtree()`.

```
void rearr(tree t_best, int rL, int rU, boolean a)
{
 boolean impr;
 worker w;
 for(i = 2; i < #species * 2 - 1; i++){
   if(sequential){
     impr = rearrangeSubtree(t_best, i, rL, rU, a);
     if(impr) applySubsequent(t_best, i);
   }
   if(parallel){
     if(w = workerAvailable) sendJob(w, t_best, i);
     else putInWorkQueue(i);
   }
 }
 if(parallel){
   while(notAllTreesReceived){
     w = receiveTree(w_tree);
     if(likelihood(w_tree) > likelihood(t_best)) t_best = w_tree;
     if(notAllTreesSent) sendJob(w, t_best, nextInWorkQueue());
  }
 }
}
```

In the sequential case rearrangements are applied to each individual subtree `i`. If the tree improves through this subtree rearrangement `t_best` is updated accordingly, i.e. subsequent topological improvements are applied. In the parallel case subtree IDs are stored in a work queue. Obviously, the subsequent application of topological improvements during 1 rearrangement step (1 invocation of `rearr()`) is closely coupled. Therefore, we slightly modify the algorithm to break up this dependency according to the following observation: Subsequent improved topologies occur only during the first 3–4 rearrangement steps (initial optimization phase). Thereafter, the likelihood is improved only by function `optimizeList20()`. This phase requires the largest amount of computation time, especially with big alignments ($\approx$ 80% of execution time). Thus, during the initial optimization phase we send only one single subtree ID `i=2,...,#species * 2 - 1` along with the currently best tree `t_best` to each worker for rearrangements. Each worker returns the best tree `w_tree` obtained by rearranging subtree `i` within `t_best` to the master. If `w_tree` has a better likelihood than `t_best` at the master, we set `t_best = w_tree` and distribute the updated best tree to each worker along with the following work request. The program assumes that the initial optimization **phase IIa** is terminated if no subsequent improved topology has been detected during the last three rearrangement steps. In the final optimization **phase IIb**, we reduce communication costs and increase granularity by generating only $5 * \#workers$ jobs (subtree ID spans). Finally, irrespective of the current optimization phase the best 20 topologies (or $\#workers$ topologies if $\#workers > 20$) computed by each worker during one rearrangement step are stored in a local worker tree list. When all `#species * 2 - 3` subtree rearrangements of `rearr()` have been completed, each worker sends its tree list to the master. The master process merges the lists and redistributes the 20 ($\#workers$) best tree topologies to the workers for branch length optimization. When all topologies have been globally optimized the master starts the next iteration of function `optimize()`. Due to the required changes to the algorithm the parallel program is non-deterministic, since final output depends on the number of workers and on the arrival sequence of results for runs with equal numbers of workers, during the initial optimization **phase IIa**. This is due to the altered implementation of the subsequent application of topological improvements during the initial rearrangement steps which leads to a traversal of search space on different paths. The program flow of the parallel algorithm is outlined in Figure 4.

2.3. Distributed algorithm

The motivation to build a distributed seti@home-like [24] code is driven by the computation time requirements for trees containing more than 1.000 organisms and by the desire to provide inexpensive solutions for this problem which do not require supercomputers. The main design principle of the distributed code is to reduce communication costs as far as possible and accept potentially bad speedup values. The algorithm of the http-based implementation is similar to the parallel program.

Initially, a gzipped alignment file is transfered to all workers which start with the computation of a local parsimony starting tree. The parsimony tree is then returned to the master as in the parallel program. However, the parallel and distributed algorithms differ in two important aspects which reduce communication costs.

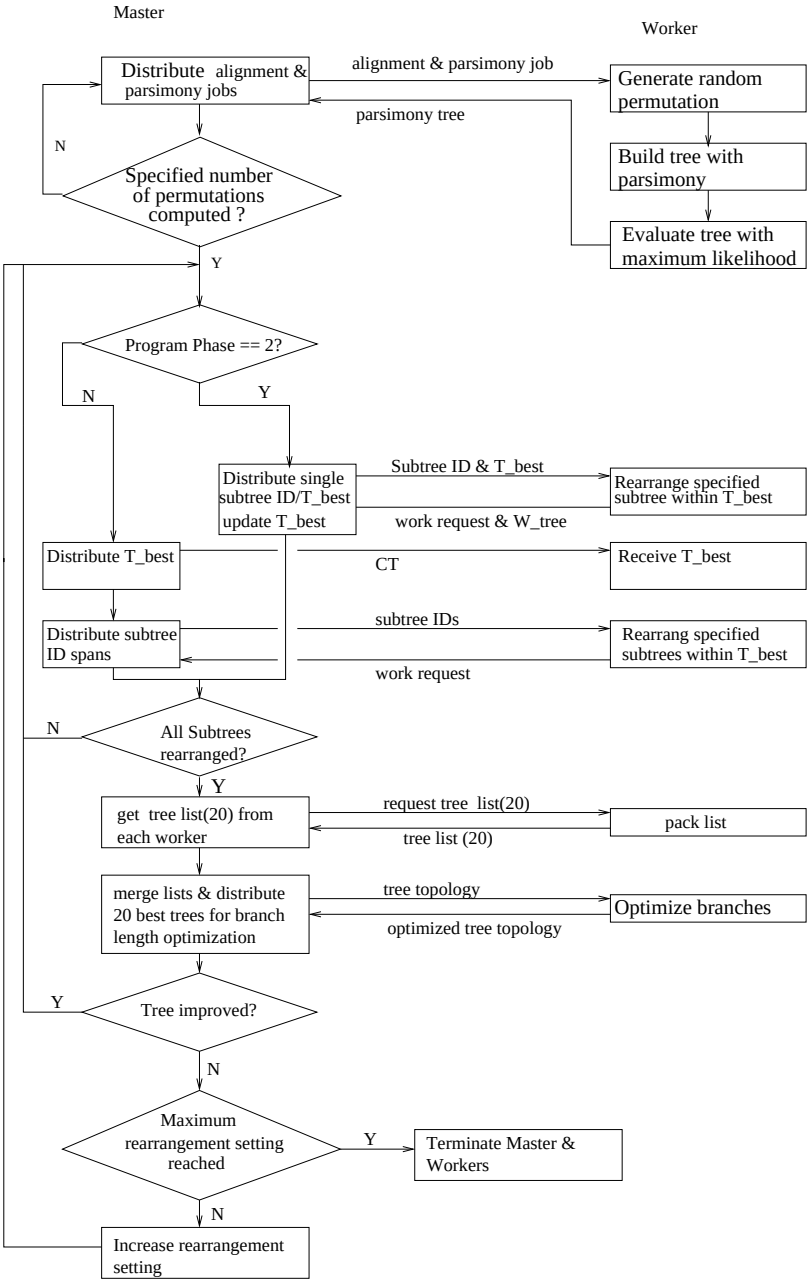


Figure 4. Parallel program flow of RAxML-II

Firstly, RAxML@home does not implement **phase IIa** but only **phase IIb** of the parallel algorithm, to avoid frequent communication and frequent exchange of tree topologies between master and workers.

Secondly, the lists containing the 20 best trees, irrespective of the number of workers, are optimized *locally* at the workers after completion of subtree rearrangements. The branch lengths of the trees in the list are optimized less exhaustively than in the sequential and parallel program. After this initial optimization only the best local tree is thoroughly optimized and returned to the master.

This induces some computational overhead and a slower improvement rate of the likelihood during the initial optimization phase (**phase IIa** of the parallel program) but remains within acceptable limits.

2.3.1. Technical issues

Some technical issues concerning the implementation of the http-based version of RAxML@home regarding communication, redundancy, and security will briefly be outlined at this point.

The communication infrastructure is provided by a http communication library. The most expensive part in terms of communication costs is the distribution of the alignment file which is compressed using `gzip`. The `gzip` shows sufficient compression rates for multiple alignments, e.g. a compression factor of 31 for a 1.000-taxon alignment.

To provide redundancy a queue with timeouts is used to ensure that every subtree rearrangement job is computed. Furthermore, failure procedures have been devised which are able to handle temporary master and worker failures.

An important security scenario is that some workers deliberately return phony trees. If the tree is not in the correct format, this can easily be detected by the routine which reads the respective tree string. The only serious security problem arises when a worker returns a tree that is in the correct format and has a "fake" likelihood, i.e. a likelihood value which is significantly better than the actual likelihood of the topology contained in the message and `t_best` at the master. In this case the likelihood of that topology is "quickly" verified by the master process. This quick verification only performs a superficial and fast branch length optimization of the tree in order to avoid excessive load of the master component. If the differences to the claimed likelihood in the tree string is $< 1\%$ the tree is accepted, otherwise it is rejected. Finally, the MD5 (Message Digest number 5) checksum is used to provide some basic authentification of messages. A detailed technical description of RAxML@home is provided in [15].

3. RESULTS

3.1. Test data, platforms & experimental setup

For our experiments we extracted alignments comprising 150, 200, 250, 500, 1.000, and 10.000 taxa (150_ARB,...,10000_ARB) from ARB containing organisms from the domains

Eukarya, Bacteria and Archaea. In addition, we used the 101 and 150 sequence data sets (101_SC, 150_SC [26]) which can be downloaded at `www.indiana.edu/~rac/hpc/fastDNAml`. Those two alignments have proved to be very hard to compute, especially for MrBayes. Furthermore, we used two well-known real data sets of 218 and 500 sequences (218_RDPII, 500_ZILLA). Finally, we used 50 synthetic 100-taxon alignments with 500bp each and the respective true reference trees which are available at `www.lirmm.fr/w3ifa/MAAS`. Details on the generation of those data sets which use e.g. varying sequence divergence rates can be found in [4]. To facilitate and accelerate testing we used the HKY85 [5]. model of sequence evolution and a fixed transition/transversion (tr/tv) ratio of 2.0 except for 150_SC (1.24) and 101_SC (1.45). All alignments including the best topologies are available for download at `wwwbode.cs.tum.edu/~stamatak`. Since the tr/tv ratio is defined differently in PHYML we scaled it accordingly for the test runs. The manual for PAML [16] contains a nice description of differences in tr/tv ratio definitions among various implementations.

The likelihood values for the final tree topologies of PHYML and RAxML-II have been computed with fastDNAml since the likelihood values for the same topology vary among programs due to numerical differences in implementations.

For real data MrBayes was executed for 2.000.000 generations using 4 MC^3 chains and the recommended random starting trees. Furthermore, we set the sample and print frequency to 5.000. To enable a fair comparison we evaluated all 400 MrBayes output trees with fastDNAml and we report the value of the topology with the best likelihood and the execution time at that point. This comparison is not necessarily fair. We use it to demonstrate that the MC^3 chains do not reach stable values (convergence) within reasonable times, i.e. less than 24 hours, for alignments comprising more than 200 sequences. For synthetic data we executed MrBayes for 100.000 generations using 4 MC^3 chains and random starting trees. We used a sample and print frequency of 500 and built a majority-rule consensus tree from the last 50 trees. Those significantly faster settings proved to be sufficient since trees for synthetic data generally converge much faster to stable values than in real data analyses.

We decided to assess performance only for those three programs since results in [30] indicate that MrBayes is the fastest and most accurate method for phylogenetic tree reconstruction, i.e. the method to beat. Furthermore, the more recently published program PHYML is -to the best of our knowledge- the fastest available sequential code for "traditional" maximum likelihood-based tree inference.

Sequential tests were conducted on a small cluster of Intel Xeon 2.4 GHz processor with 4GB of main memory. All programs were compiled using `icc -O3` (native Intel compiler).

For parallel performance analysis RAxML-II was executed on the 2.66GHz Intel Xeon cluster at Regionales Rechenzentrum Erlangen (RRZE [20]) on 1, 4, 8, 16, and 32 processors.

In all cases RAxML-II was executed with the standard rearrangement parameter setting, i.e. `rStart = 5` and `rMax = 21`.

3.2. Sequential tests

In Table I we summarize the final likelihood values and execution times in seconds obtained with PHYML, MrBayes, and RAxML-II for real data sets. The results listed for RAxML-II correspond to the best of 10 runs with distinct randomized parsimony starting trees. In

Table I. PHYML, MrBayes, RAxML-II execution times and likelihood values for real data sets

data	PHYML	secs	MrBayes	secs	RAxML	secs	R > PHY	secs
101_SC	-74097.6	153	-77191.5	40527	-73919.3	617	-74046.9	31
150_SC	-44298.1	158	-52028.4	49427	-44142.6	390	-44262.9	33
150_ARB	-77219.7	313	-77196.7	29383	-77189.7	178	-77197.6	67
200_ARB	-104826.5	477	-104856.4	156419	-104742.6	272	-104809.0	99
250_ARB	-131560.3	787	-133238.3	158418	-131468.0	1067	-131549.4	249
500_ARB	-253354.2	2235	-263217.8	366496	-252499.4	26124	-252986.4	493
1000_ARB	-402215.0	16594	-459392.4	509148	-400925.3	50729	-401571.9	1893
218_RDPII	-157923.1	403	-158911.6	138453	-157526.0	6774	-157807.9	244
500_ZILLA	-22186.8	2400	-22259.0	96557	-21033.9	29916	-22036.9	67

addition, since execution times of RAxML-II might appear long compared to PHYML in column R > PHY we indicate the likelihood and the time at which RAxML-II detects a topology with a better likelihood value than the final PHYML tree.

For sake of completeness we also indicate the number of base pairs (bp), worst results and worst execution times obtained with RAxML-II for each data set in a separate Table II. In the last two columns of this Table we list the final likelihood values and execution times in hours (!) obtained for the same data sets with PAxML. As already mentioned, PAxML is exactly equivalent to parallel fastDNAml, but faster by approximately 50%. The results were obtained from parallel runs on the HeLiCs [6] PC cluster and the highest feasible rearrangement setting, in terms of acceptable computation times. The enormous improvement of execution times illustrates the algorithmic progress in the field over the last two years.

The long overall execution times of RAxML-II compared to PHYML are due to the asymptotic convergence of likelihood over time which is typical for the tree optimization process. A particularly extreme example for this type of convergence behavior is illustrated in Figure 5 for 500_ZILLA. Therefore, the comparatively small differences in final likelihood values which are usually below 1% should not be underestimated, in terms of the computational effort required to obtain those values. In addition, the application of the Kishino-Hasegawa likelihood ratio test [11] shows that the topologies obtained by RAxML-II are significantly better than those of PHYML.

Finally, in Figure 6 we plot the relative topological accuracy (Robinson Foulds-rate [21]) of PHYML, RAxML-II, and MrBayes for 50 100-taxon trees which are enumerated on the x-axis. The average RF-rate for PHYML is 0.0796, 0.0808 for RAxML-II, 0.0818 for RAxML-II with a less exhaustive rearrangement parameter setting, and 0.0741 for MrBayes. The average execution time of RAxML-II was 131.05 seconds and 29.27 seconds for the less exhaustive search. PHYML required an average of 35.21 seconds and MrBayes 945.32 seconds. The experiments suggest that there is no significant difference between PHYML and RAxML-II for synthetic data in contrast to the results obtained with real data.

Table II. Alignment lengths, worst performance data from 10 RAxML-II runs, and PAxML performance data

data	bp	RAxML-II	secs	PAxML	hrs
101_SC	1858	-73982.42	1021	-73975.9	47
150_SC	1269	-44159.89	467	-44146.9	164
150_ARB	3188	-77198.98	305	-77189.8	300
200_ARB	3270	-104743.32	1236	-104743.3	775
250_ARB	3638	-131513.04	1758	-131469.0	1947
500_ARB	4030	-252631.93	26124	-252588.1	7372
1000_ARB	5547	-401006.52	66902	-402282.1	9898
218_RDPII	4182	-157580.21	7432	n/a	n/a
500_ZILLA	759	-21087.46	29916	n/a	n/a

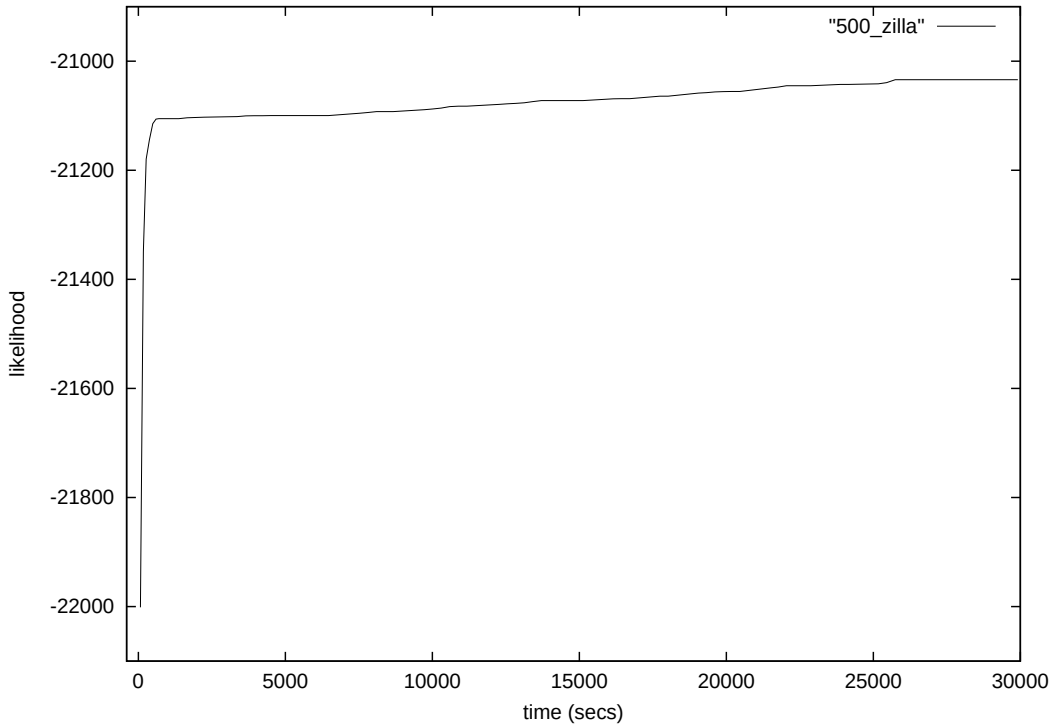


Figure 5. Likelihood improvement over time of RAxML-II for 500_ZILLA

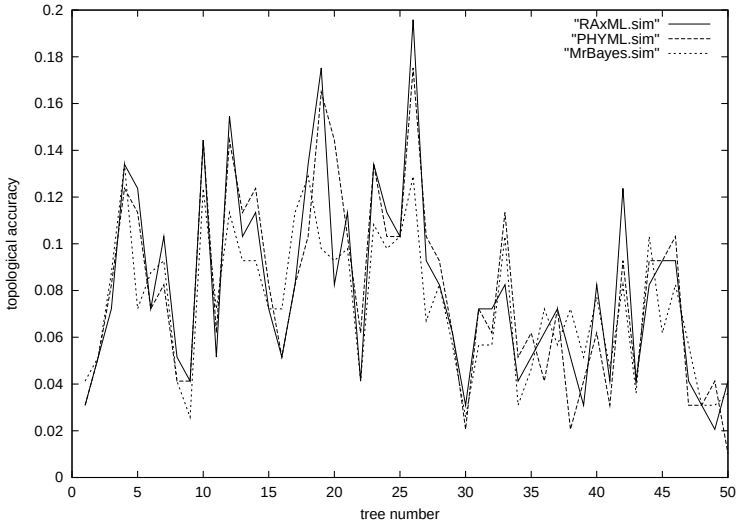


Figure 6. Topological accuracy of PHYML, RAxML-II and MrBayes for 50 100-taxon trees

In Figures 7 (101_SC) and 8 (500_ARB) we plot MrBayes likelihood values over generation numbers for runs with RAxML-II and random starting trees. These figures illustrate how starting trees obtained by RAxML-II can be deployed to accelerate convergence of MC^3 analyses. In order to demonstrate the rapid tree optimization capabilities of RAxML-II in Figures 9 and 10 we plot the likelihood improvement over time of RAxML-II and MrBayes for the same 150_SC and 150_ARB random starting trees. The final likelihood values obtained by RAxML-II for those runs were -44149.18 (150_SC) and -77189.78 (150_ARB) respectively. Figure 7 also underlines one of the main problems of MC^3 analysis. The same problem is also pointed out by Huelsenbeck in [10]: When to stop the chain? In this examples the bayesian inference with a random starting tree seems to have reached apparent stationarity. However, the likelihood is significantly inferior to the likelihood of the chain with the RAxML-II starting tree. The same behavior has also been observed for the 150_SC data set. Thus, "good" user trees are a useful reference and can significantly accelerate bayesian analysis. This justifies the work on fast "traditional" maximum likelihood methods after the emergence and great impact of bayesian methods [9]. Thus, we do not consider RAxML-II as concurrence to MrBayes, but rather as useful tool to improve bayesian inference and vice versa.

3.3. Parallel tests

We conducted parallel speedup tests with a fixed starting tree for 1000_ARB. The program was executed on 1, 4, 8, 16, and 32 processors. To calculate the speedup values we only take into account the number of workers, since the master process hardly produces any load. In Figure 11

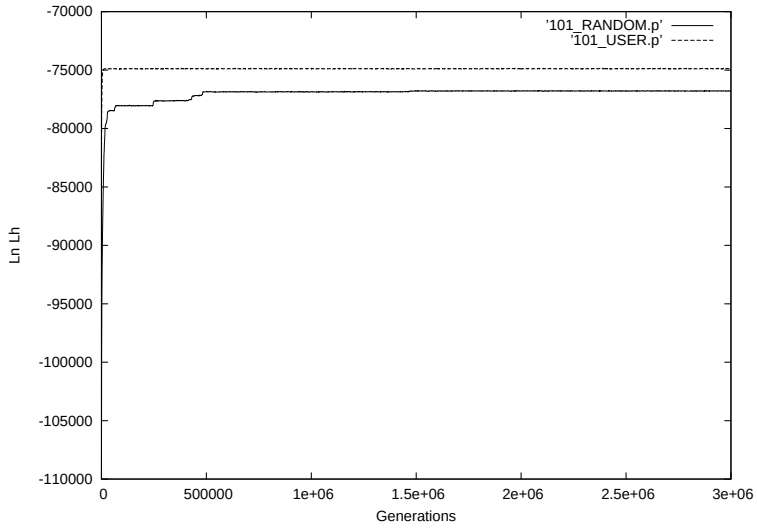


Figure 7. Convergence behavior of MrBayes for 101_SC with user and random starting trees over 3.000.000 generations

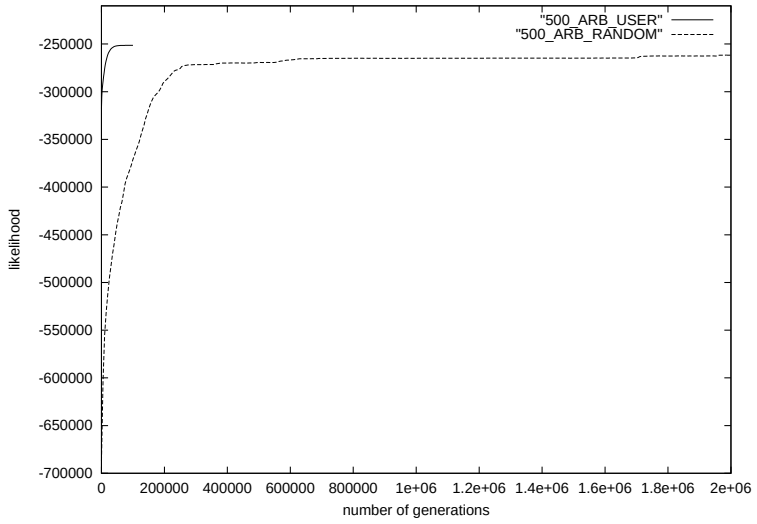


Figure 8. Convergence behavior of MrBayes for 500_ARB with user and random starting trees over 2.000.000 generations

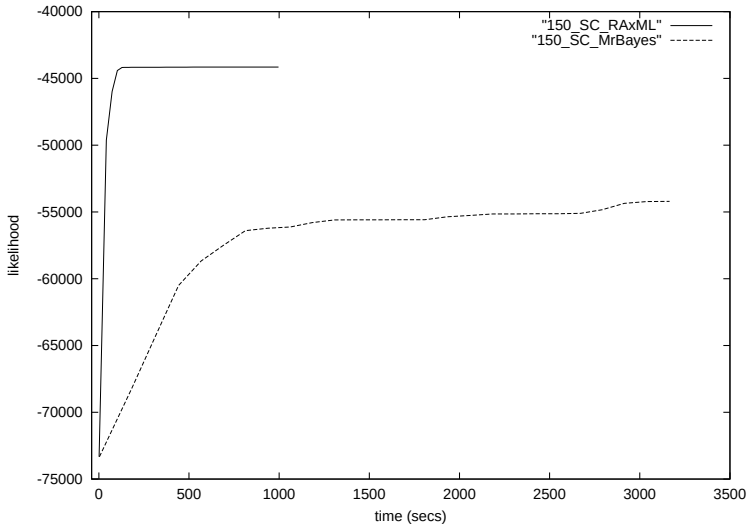


Figure 9. 150_SC likelihood improvement over time of RAxML-II and MrBayes for the same random starting tree

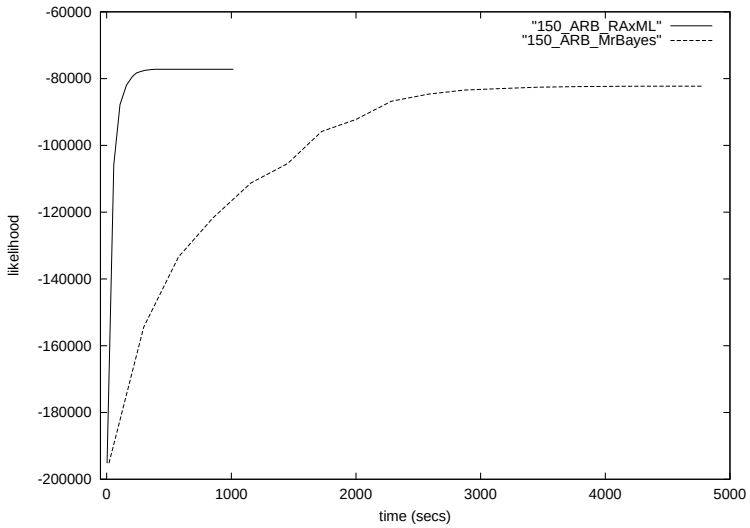


Figure 10. 150_ARB likelihood improvement over time of RAxML-II and MrBayes for the same random starting tree

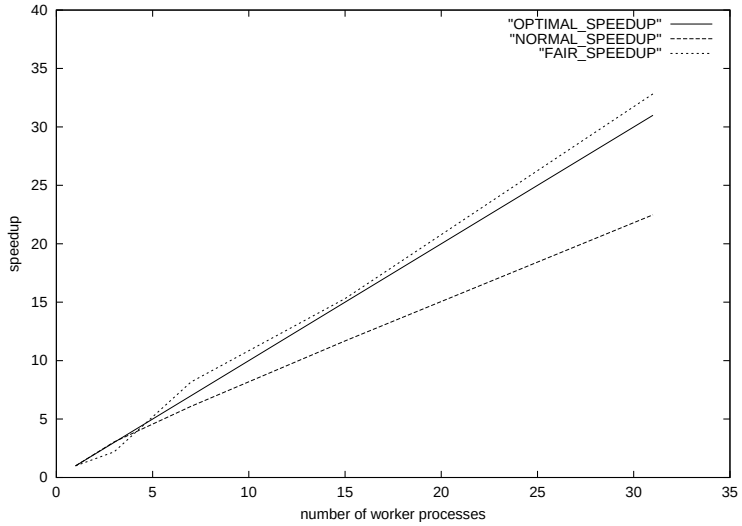


Figure 11. Normal, fair, and optimal speedup values for 1000_ARB with 3,7,15, and 31 worker processes on the RRZE PC Cluster

we plot "fair" and "normal" speedup values obtained for the experiments with the 1000_ARB data set to account for the non-determinism of the program. "Fair" speedup values indicate the first point of time at which the parallel code encounters a tree with a better likelihood than the final tree of the sequential run or vice versa. These "fair" values better correspond to real program performance. Furthermore, we also indicate "normal" speedup values which are based on the entire execution time of the parallel program, irrespective of final likelihood values. Since we intend to explore the effect of non-determinism on program performance we executed the parallel code 4 times for each job-size and calculated average "normal"/"fair" execution times and likelihood values. The experiments demonstrate that the non-determinism of the parallel algorithm does not have a negative impact on program performance.

3.4. Inference of a 10.000-taxon tree

The computation of the 10.000-taxon tree was conducted using the sequential, as well as the parallel version of RAxML. One of the advantages of RAxML-II consists in the randomized generation of starting trees. Thus, we computed 5 distinct randomized parsimony starting trees sequentially along with the first 3–4 rearrangement steps on a small cluster of Intel Xeon 2.4GHz processors at our institute. This phase required an average of 112.31 CPU hours per tree. Thereafter, we executed several subsequent parallel runs (due to job run-time limitations of 24 hrs) with the respective starting trees on either 32 or 64 processors at the RRZE 2.66GHz Xeon-cluster. The parallel computation required an average of 1689.6 accumulated CPU hours

per tree. The best likelihood for 10000_ARB was -949570.16 the worst -950047.78 and the average -949867.27. PHYML reached a likelihood value of -959514.50 after 117.25 hrs on the Itanium2. Note, that the parsimony starting trees computed with RAxML-II had likelihood values ranging between -954579.75 and -955308.00. The average time required for computing those starting trees was 10.99 hrs.

4. DISCUSSION

We have presented simple heuristics and efficient implementations for sequential, parallel, and distributed maximum likelihood-based phylogenetic tree inference. The sequential algorithm outperforms the currently -to the best of our knowledge- fastest and most accurate programs for phylogenetic tree inference on real-world data. Tree inference for synthetic data sets using RAxML-II is equally accurate as with PHYML. MrBayes has shown to be slightly more accurate for synthetic data than RAxML-II and PHYML but is significantly slower. Moreover, we have shown that for some real data sets MrBayes does not converge in reasonable times or has reached apparent stationarity while the likelihood values of the chain are significantly inferior to those obtained by "traditional" maximum likelihood searches. The sequential, parallel, and distributed source code of RAxML-II including all test data sets and final tree topologies is freely available at: `wwwbode.cs.tum.edu/~stamatak`.

Another important performance criterion regarding inference of huge trees consists in the memory consumption of the three analyzed programs. For example RAxML-II consumed 199MB, PHYML 880MB, and MrBayes 1.195MB of main memory for 1000_ARB. Furthermore, both MrBayes and PHYML exited with error messages due to excessive memory requirements for the 10.000 taxon alignment on the Xeon 2.4GHz processors. Therefore, we made an effort to port MrBayes and PHYML to a 64-bit Intel Itanium2 1.3GHz processor with 8GB of main memory. While MrBayes exited for unknown reasons,PHYML finally required 8.8GB of main memory. Note, that RAxML-II used only 800MB for the 10.000 taxon alignment.

The parallel inference of a 10.000-taxon tree has revealed some well-known problems and challenges concerning inference of huge trees [23]. In particular, visualization and quality assessment of large trees are problematic since there exist no appropriate visualization tools and bootstrapping techniques are not applicable due to the high computational cost.

Current work on RAxML-II focuses on implementation of more elaborate models of nucleotide substitution and model parameter estimation, as well as on an implementation for protein sequence data. In addition, we currently work on RAxML-TreeDivisor, a tree-based alignment subdivision tool for divide-and-conquer maximum likelihood supertree approaches [22].

Future work will cover the implementation of a hybrid algorithm which will be based on the RAxML-II tree optimization heuristics for the initial optimization phase and simulated annealing and related sampling techniques for the final optimization phase.

REFERENCES

1. Brauer MJ, Holder MT, Dries LA, Zwickl DJ, Lewis PO, Hillis DM. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Molecular Biology and Evolution* 2002; (19): 1717–1726.
2. Felsenstein J. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution* 1981; (17): 368–376.
3. Feng X, Buell DA, Rose JR, Waddell PJ. Parallel algorithms for Bayesian phylogenetic inference. *Journal of Parallel and Distributed Computing: Special Issue on High-Performance Computational Biology* 2003; (63): 707–718.
4. Guindon S, Gascuel O. A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Systematc Biology* 2003; **52**(5): 696–704.
5. Hasegawa M, Kishino H, Yano T. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolutio* 1985; (22): 160–174.
6. HEidelberg LInux Cluster System homepage: helics.uni-hd.de [15 March 2003]
7. Holder MT, Lewis PO. Phylogeny Estimation: Traditional and Bayesian Approaches. *Nature Reviews Genetics* 2003; (4): 275–284.
8. Huelsenbeck JP, Ronquist F. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 2001; **17**(8): 754–5.
9. Huelsenbeck JP, Ronquist F, Nielsen R, Bollback JP. Bayesian inference and its impact on evolutionary biology. *Science* 2001; (294): 2310–2314.
10. Huelsenbeck JP, Larget B, Miller RE, Ronquist F. Potential Applications and Pitfalls of Bayesian Inference of Phylogeny. *Systematic Biology* 2002; **51**(5): 673–688.
11. Kishino H, Hasegawa M. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data and the branching order in Homonoidae. *Journal of Molecular Evolution* 1989; (29): 170–179.
12. Lemmon A, Milinkovitch M. The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. *Proceedings of the National Academy of Sciences USA* 2002; (99): 10516–10521.
13. Ludwig W et al. ARB: a software environment for sequence data. *Nucleic Acids Research* 2004; **32**(4): 1363–1371.
14. Olsen GJ, Matsuda H, Hagstrom R, Overbeek R. fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *Computer Applications in the Biosciences* 1994; (10): 41–48.
15. Ott M. PAxML@home: Speceification and development of a globally distributed software architecture for computation of phylogenetic trees. *Technische Universität München, Master's thesis* 2004;
16. PAML Manual (Information on tr/tv definitions: page 20): bcr.musc.edu/manuals/pamlDOC.pdf [11 November 2003]
17. parallel fastDNAml downlaod site: www.indiana.edu/~rac/hpc/fastDNAml [12 February 2003]
18. PAUP project site: paup.csit.fsu.edu [30 May 2003]
19. PHYLIP homepage. evolution.genetics.washington.edu/phylip.html [14 April 2003]
20. Regionales Rechenzentrum Erlangen: HPC services. www.rrze.uni-erlangen.de [12 October 2004]
21. Robinson D, Foulds L. Comparison of weighted labeled trees. *Lecture Notes in Mathematics* 1979; (748): 119–126, Springer, Berlin.
22. Roshan U, Moret BME, Williams TL, Warnow T. Performance of supertree methods on various data set decompositions. *Bininda-Edmonds ORP (editor) Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life* to be published; 301–328. Preprint available at www.cs.unm.edu/~tlw/publications.html.
23. Sanderson MJ, Driskell AC. The challenge of constructing large phylogenetic trees. *Trends in Plant Science* 2003; **8**(8): 374–378.
24. Seti@home project site: setiathome.ssl.berkeley.edu [20 July 2003]
25. Stamatakis AP, Ludwig T, Meier H, Wolf MJ. November 2002. Accelerating Parallel Maximum Likelihood-based Phylogenetic Tree Computations using Subtree Equality Vectors. *Proceedings of 15th IEEE/ACM Supercomputing Conference (SC2002)* 2002; CD proceedings.
26. Stewart CA, Hart D, Berry DK, Olsen GJ, Wernert E, Fischer W. Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference. *Proceedings of 14th IEEE/ACMSupercomputing Conference (SC2001)* 2001; CD proceedings.

27. Strimmer K, Haeseler Av. Quartet Puzzling: A Maximum-Likelihood Method for Reconstructing Tree Topologies. *Molecular Biology and Evolution* 1996; (13): 964–969.
28. Schmidt HA et al. TREE-PUZZLE: Maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics* 2002; (18): 502–504.
29. Tuffley C, Steel M. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. *Bulletin of Mathematical Biology* 1997; (3): 581–607.
30. Williams TL, Moret BME. An Investigation of Phylogenetic Likelihood Methods. *Proceedings of 3rd IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03)* 2003; CD proceedings.