# New Fast and Accurate Heuristics for Inference of Large Phylogenetic Trees *

Alexandros P. Stamatakis, Harald Meier
Technische Universität München, Department of Computer Science
Boltzmannstr. 3, D-85748 Garching b. München, Germany
{stamatak,meierh}@cs.tum.edu

Thomas Ludwig
Ruprecht-Karls-University, Department of Computer Science
Im Neuenheimer Feld 348, D-69120 Heidelberg, Germany
thomas.ludwig@informatik.uni-heidelberg.de

## Abstract

*Inference of phylogenetic trees comprising hundreds or even thousands of organisms based on the maximum likelihood method is computationally extremely expensive. We present simple new heuristics which yield accurate trees for synthetic (simulated) as well as real data and significantly reduce execution time. The new heuristics have been implemented in a program called RAxML which is freely available as open source code. Furthermore, we present a distributed version of our algorithm which is implemented in an MPI-based prototype. This prototype is currently being used to implement an http-based seti@home-like version of RAxML. We compare our program with PHYML and MrBayes which to our best 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 real-world alignments comprising 101 up to 1000 sequences. RAxML outperforms MrBayes for real-world data both in terms of speed and final likelihood values. Furthermore, for real data RAxML 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 and PHYML but significantly slower.*

*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".

## 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 [1]. 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 [7] database in distributed and parallel environments. In previous work [16] 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 our concept in parallel fastDNAml [8, 17] 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 very 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 (Randomized Axelerated Maximum Likelihood) as well as in a parallel MPI-based prototype for the distributed http-based version of our code RAxML@home.

The new RAxML-release presented here is a significantly changed and improved version of the initial RAxML algorithm we describe in [14, 15].

The remainder of this paper is organized as follows:

In Section 2 we briefly describe related work with a focus on current state-of-the-art programs for maximum likelihood-based and bayesian phylogenetic infer-

ence, which we use to assess the quality of RAxML. In the following Section 3 we describe the new heuristics and algorithm. The design of the distributed algorithm is outlined in Section 4 whereas in Section 5 we summarize our experimental results. Finally in Section 6 we conclude and briefly address current and future issues of our work.

## 2. 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 maintained by J. Felsenstein [11] lists most available programs. On the other hand there exist bayesian methods which are relatively new compared to maximum likelihood and have experienced great impact, especially through the release of a program called MrBayes [5].

A thorough comparison of popular phylogeny programs using statistical approaches such as fastDNAml, MrBayes, PAUP [10], and treepuzzle [18] based on synthetic data has been conducted by T.L. Williams et al [19]. 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 Markov Chain Monte Carlo (MCMC) technique.

Recently, Guidon and Gascuel published an interesting paper about their new program PHYML [2], 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 optimal tree in respect to the likelihood value and like MrBayes is also capable of optimizing model parameters.

Thus, to our best knowledge MrBayes and PHYML are currently the fastest and most accurate representatives of bayesian and "traditional" approaches to phylogenetic tree inference. Therefore, we focus on those two programs for assessing performance of RAxML.

One should however be careful when comparing bayesian with maximum likelihood methods due to subtle differences in the statistical models (an introduction to bayesian phylogenetic inference can be found in [4]). This is due to the fact that bayesian methods tend to optimize topologies in respect to a broader range of model parameters, whereas maximum likelihood methods optimize the likelihood of topologies with usually fixed or restricted model parameters. Thus, a bayesian analysis might not yield the peak likelihood values as obtained from a maximum likelihood search.

## 3. New Heuristics

"Traditional" maximum likelihood searches can be implemented in two ways: On the one hand they can start from scratch and insert organisms progressively into the tree such as the stepwise addition algorithm (implemented e.g. in [1, 8, 21]). On the other hand they can start with an initial tree already containing all organisms built by a simpler method such as Neighbor Joining (NJ) or by random (implemented in [2, 5]). The likelihood of such a starting tree is then progressively optimized by application of minor topological changes.

RAxML belongs to this second class of algorithms. The first part of our heuristics consists in building a starting tree using dnapars from PHYLIP [11] for two reasons:

*Firstly*, parsimony is related to maximum likelihood under simple evolutionary models [20], such that we can expect to obtain a starting tree with a relatively good likelihood value compared to random or NJ starting trees. For example the 500_ZILLA parsimony starting tree had already a better likelihood than the final tree of PHYML as depicted in Section 5, Table 1.

*Secondly*, dnapars uses stepwise addition for tree building and is relatively fast. This enables the construction of different starting trees by using a randomized input sequence ordering, since distinct input orderings produce distinct final trees. Thus, RAxML can be run several times with different starting trees and the set of final trees may be used for building a consensus tree and augment confidence in the final result.

We removed however some optimization steps from the dnapars algorithm to accelerate computations.

The second and most important part of our heuristics is the tree optimization process. RAxML performs simple tree rearrangements by subsequently removing all possible subtrees from the present tree and inserting them into neighbouring branches up to a specified distance of nodes. RAxML 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 (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). The likelihood of the such generated topologies is evaluated and the best tree is kept. If one alternative topology improves the likelihood the process is repeated with the new tree until no better topology is found.

The rearrangement process of RAxML 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 Figure 1 with bold lines RAxML

only optimizes the three branches adjactent to the insertion point 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 good alternative trees. RAxML uses a list of only size 20 to store the best trees obtained during one rearrengement step. This list size proved to be a practical value in terms of speed and thoroughness of the search. The algorithm performs global branch length optimizations only on those 20 best trees after completion of each rearrangement step. Due to the capability to analyze many more alternative topologies in less time higher rearrangements settings can be used e.g. 1–5 or 1–10 which results in significantly improved final trees.

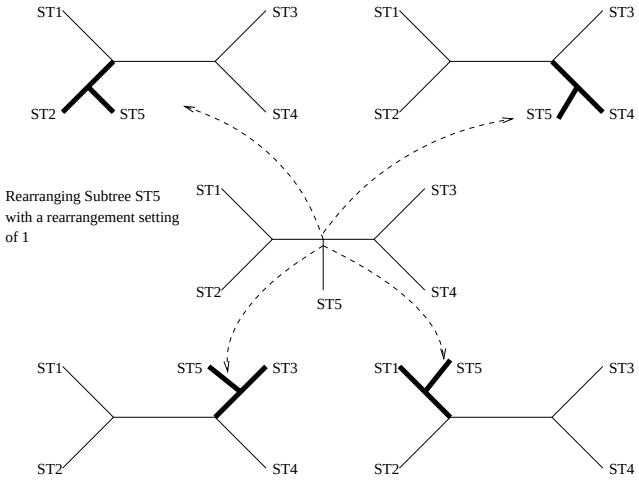


**Figure 1. Rearrangments traversing one node for subtree ST5, branches which are optimized are indicated by bold lines**

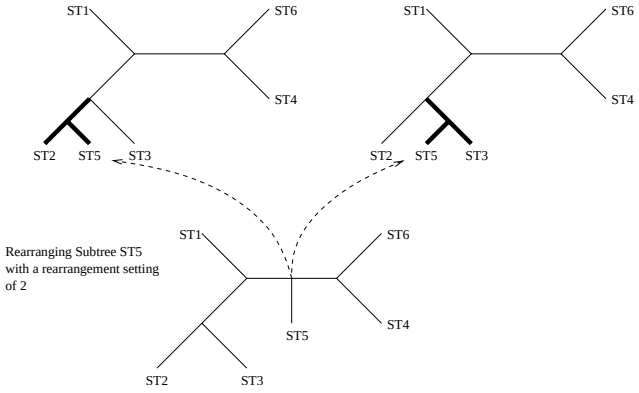


**Figure 2. Example rearrangments traversing two nodes for subtree ST5, branches which are optimized are indicated by bold lines**

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 found this tree is kept immediatly and all subsequent subtree rearrangements are performed on the improved topology. This enables rapid optimization of random starting trees as depicted e.g. for two alignemnts containing 150 taxa in Figures 9 and 10 (a description of those data sets can be found in Section 5).

## 4. Distributed Algorithm

Our Motivation to build a distributed seti@home-like [13] code is driven by the computation time requirements for trees containing more than 1000 organisms and by the desire to provide inexpensive solutions for this problem which do not require supercomputers. The main design principle of our distributed code is to reduce communication costs as far as possible and accept potentially bad speedup-values. The protoype implementation is based on a simple master-worker architecture and consists of two phases.

In phase I our distributed algorithm starts by distributing the alignment file to all worker processes. The alignment data transfer is not critical since alignments show good compression ratios with gzip which forms part of our http communication library. We attained a compression by factor 31 for the 1000 taxa alignment mentioned in Section 5. Thereafter, each worker independently computes a randomized parsimony starting tree and sends it to the master process.

In phase II the master initiates the optimization process for the best parsimony starting tree. Due to the high speed of a single topology evaluation it is not feasible to distribute work by topologies as e.g. in parallel fastDNAml. Instead we distribute work by sending a span of subtree node numbers i.e. IDs for the subtrees which shall be moved along with the currently best topology to the workers. At present we generate $2 * \#workers$ jobs for load-balancing. The maximum amount of jobs that can be created in this way is approximately equal to the number of taxa. The worker then performs rearrangements for the specified subtrees within its currently best tree and returns the best tree to the master. The best topologies from this process are stored in a local worker tree list with 20 entries. When all subtree rearrangements of one rearrangement step have been calculated, each worker performs a branch-length optimization of its local list containing the 20 best trees. Finally each worker returns its currently best tree to the master and the process of subtree ID distribution and evaluation of local best lists continues until no better tree is found.

Due to the required changes to the algorithm the distributed program does not yield exactly the same results as the sequential program for a fixed starting tree. However, we might expect slightly better results since a larger number of trees is branch-length optimized ($\#workers * 20$). Final results also depend on the number of workers and might vary from run to run. Since the subsequent application of topological improvements is closely coupled we have to accept a slower convergence during the initial phase of the computation. Our experiments have shown that multiple improved topologies are detected only during the first 3–4 rearrangement steps. Thereafter, only one alternative topology per rearrangement step improves the likelihood. This observation is illustrated in Figure 3 where we plot the number of improved topologies per rearrangement step for a 150_SC (a description of the data we used is provided in Section 5) tree inference with a random and a parsimony starting tree. When the number of improved topologies is zero the improved tree has been obtained by optimizing a toplogy of the best tree list.

An MPI-based parallel version of our program can easily be derived from the distributed prototype by maintaining a single, central tree list at the master process and distributing the best 20 topologies for branch length optimization to the workers.

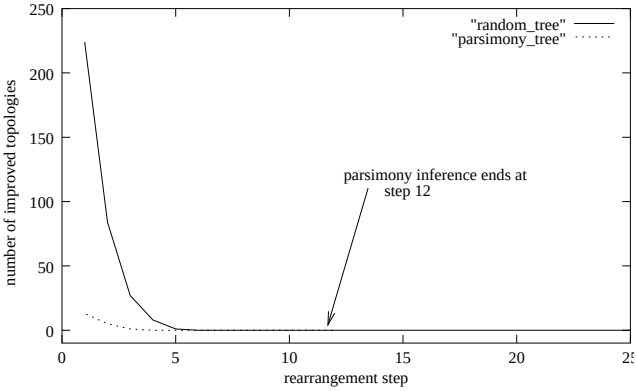


**Figure 3. Number of improved topologies per rearrangement step for a SC_150 random and parsimony starting tree**

## 5. Results

For our experiments we extracted alignments comprising 150, 200, 250, 500 and 1000 taxa (150_ARB,...,1000_ARB) from the ARB [7] small subunit ribosomal ribonucleic acid (ssu rRNA) database containing organisms from the domains Eucarya, Bacteria and Archaea. In addition, we used the 101 and 150 sequence data sets (101_SC, 150_SC [17]) which can be downloaded at `www.indiana.edu/~rac/hpc/fastDNAml` and have proved to be very hard to compute, especially for MrBayes. In addition 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 [2]. To facilitate and accelerate testing we used the HKY (Hasegawa et al. 1985) model of sequence evolution and a 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 together with the RAxML source code at `wwwbode.cs.tum.edu/~stamatak`.

Since the transition/transversion ratio is defined differently in PHYML we scaled it accordingly for the test runs (the manual for PAML [9] contains a nice description of differences in the Tr/Tv ratio definitions). MrBayes does not provide a possibility to set the Tr/Tv ratio to a specific value such that we had to let the program optimize this value. We did however not observe differences in the order of final RAxML, PHYML, and MrBayes likelihood values for different Tr/Tv settings. This is illustrated in Figure 4 for the final 150_SC topologies.

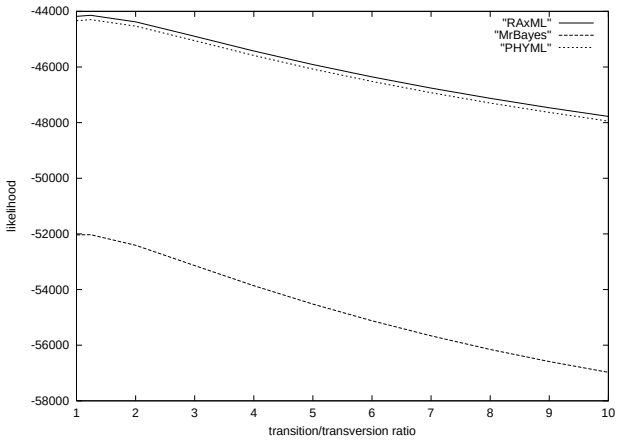


**Figure 4. RAxML, PHYML, and MrBayes final likelihood values over transition/transversion ratios for 150_SC**

For real data MrBayes was executed for 2.000.000 generations using 4 Metropolis-Coupled MCMC ($MC^3$) chains and the recommended random starting trees. Furthermore we used a sample and print frequency of 5000. To enable a fair comparison we evaluated all 400 output trees with fastDNAml and we report the value of the topology with the best likelihood and the execution time at that point. For syn-

thetic data we executed MrBayes for 100.000 generations using 4 MCMC chains and random starting trees. We used sample and print frequencies 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 converged much faster than trees for real data in our experiments.

We decided to assess performance only for those three programs since results in [19] 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 our best knowledge, the fastest available sequential code for "traditional" maximum likelihood-based tree inference.

We stress the importance of using several real data alignements since differences between phylogeny programs can often only be observed with real data.

**Sequential Tests:** All sequential tests were performed on an Intel Xeon 2.4 GHz Processor. We compiled the programs using `icc -O3` (native Intel compiler).

In Table 1 we summarize the final likelihood values and execution times in seconds obtained with PHYML, MrBayes, and RAxML. The results listed for RAxML correspond to the best of 10 runs with different randomized parsimony starting trees. For sake of completeness we also indicate the number of base pairs (bp), worst results and worst execution times obtained with RAxML for each data set in a seperate Table 2.

In addition, since execution times of RAxML might seem long compared to PHYML in column $R > PHY$ we indicate the likelihood and the time at which RAxML passed the final likelihood obtained by PHYML for a distinct series of RAxML runs. Finally, in the last two columns we list the final likelihood values and execution times in hours (!) obtained with PAxML (exactly equivalent to parallel fastDNAml, but faster by $\approx$ 50%). The results were obtained from parallel runs on the HeLiCs [3] PC cluster and the highest feasible rearrangement setting, in terms of acceptable computation times. The enormous improvement of execution times illsutrates the progress in the field over the last two years.

The long overall execution times of RAxML 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 behaviour is illustrated in Figure 5 for 500_ZILLA. Therefore, the comparatively small differences in final likelihood values wich are usually below 1% should not be underestimated, in terms of the computational effort required to obtain those values.

Finally, in Figure 6 we plot the topological accuracy (Robinson-Foulds rate) of PHYML, RAxML, and MrBayes

| data | bp | RAxML | secs |
|---|---|---|---|
| 101_SC | 1858 | -73982.42 | 1021 |
| 150_SC | 1269 | -44159.89 | 467 |
| 150_ARB | 3188 | -77198.98 | 305 |
| 200_ARB | 3270 | -104743.32 | 1236 |
| 250_ARB | 3638 | -131513.04 | 1758 |
| 500_ARB | 4030 | -252631.93 | 26124 |
| 1000_ARB | 5547 | -401006.52 | 66902 |
| 218_RDPII | 4182 | -157580.21 | 7432 |
| 500_ZILLA | 759 | -21087.46 | 29916 |

**Table 2. Worst execution times and likelihood values for real data from 10 RAxML runs**

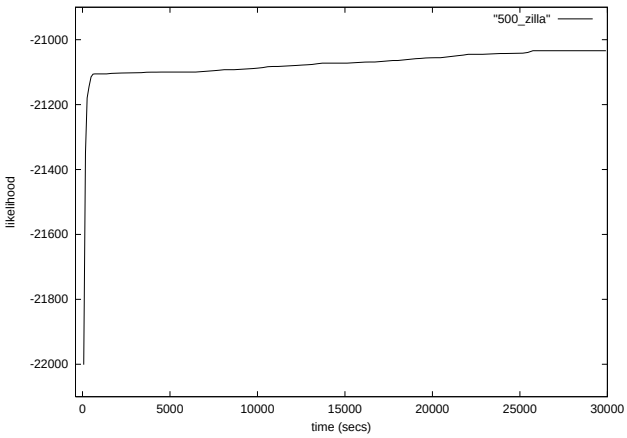


**Figure 5. Likelihood improvement over time of RAxML for 500_ZILLA**

for 50 100-taxon trees which are enumerated on the x-axis. The average R-F rate for PHYML is 0.0796, 0.0808 for RAxML, 0.0818 for RAxML with a less exhaustive search and 0.0741 for MrBayes. The average execution time of RAxML 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 illustrate that there seems to be no significant difference between PHYML and RAxML for synthetic data in contrast to the results obtained with real data. Thus, real as well as synthetic data should be used to perform comparative analyses of phylogeny programs.

Two examples which underline this argument are depicted in Figures 7 and 8 for the 101_SC and 500_ARB alignments respectively. We plot the MrBayes likelihood values over generation numbers for runs with RAxML and random starting trees. Figures 7 and 8 also illustrate the main problem of MCMC analysis which is also pointed out by Huelsenbeck in [6]: When to stop the chain? In both examples the run with random starting trees seems to have

| data | PHYML | secs | MrBayes | secs | RAxML | secs | R > PHY | secs | PAxML | hrs |
|---|---|---|---|---|---|---|---|---|---|---|
| 101_SC | -74097.6 | 153 | -77191.5 | 40527 | -73919.3 | 617 | -74046.9 | 31 | -73975.9 | 47 |
| 150_SC | -44298.1 | 158 | -52028.4 | 49427 | -44142.6 | 390 | -44262.9 | 33 | -44146.9 | 164 |
| 150_ARB | -77219.7 | 313 | -77196.7 | 29383 | -77189.7 | 178 | -77197.6 | 67 | -77189.8 | 300 |
| 200_ARB | -104826.5 | 477 | -104856.4 | 156419 | -104742.6 | 272 | -104809.0 | 99 | -104743.3 | 775 |
| 250_ARB | -131560.3 | 787 | -133238.3 | 158418 | -131468.0 | 1067 | -131549.4 | 249 | -131469.0 | 1947 |
| 500_ARB | -253354.2 | 2235 | -263217.8 | 366496 | -252499.4 | 26124 | -252986.4 | 493 | -252588.1 | 7372 |
| 1000_ARB | -402215.0 | 16594 | -459392.4 | 509148 | -400925.3 | 50729 | -401571.9 | 1893 | -402282.1 | 9898 |
| 218_RDPII | -157923.1 | 403 | -158911.6 | 138453 | -157526.0 | 6774 | -157807.9 | 244 | n/a | n/a |
| 500_ZILLA | -22186.8 | 2400 | -22259.0 | 96557 | -21033.9 | 29916 | -22036.9 | 67 | n/a | n/a |

**Table 1. PHYML, MrBayes, RAxML execution times and likelihood values for real data sets**

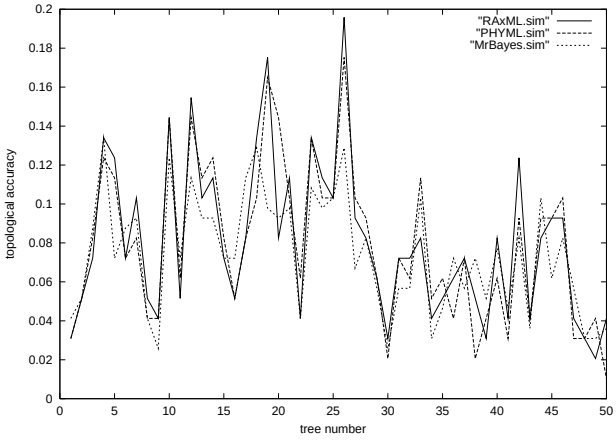


**Figure 6. Topological accuracy of PHYML, RAxML and MrBayes for 50 100-taxon trees**

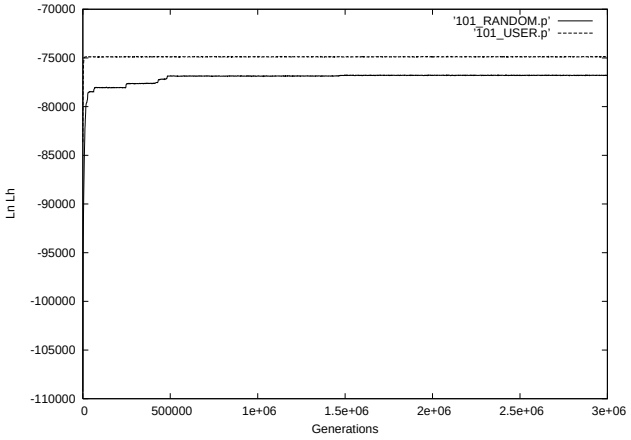


**Figure 7. Convergence behaviour of MrBayes for 101_SC with user and random starting trees over 3.000.000 generations**

reached apparent stationarity. Furthermore, they show that "good" user trees can be very useful both as reference and starting trees as well as to significantly accelerate computations. This justifies the work on fast "traditional" maximum likelihood methods after the emergence and great impact of bayesian methods. Thus, we do not see RAxML as concurrence to MrBayes, but rather as useful tool to improve bayesian inference and vice versa. Therefore, in order to facilitate the analysis process RAxML produces an output file containing the alignment and the final tree in MrBayes input format.

Finally, in order to demonstrate the rapid tree otimization capabilities of RAxML in Figures 9 and 10 we plot the likelihood improvement over time of RAxML and MrBayes for the same 150_SC and 150_ARB random starting trees. The final likelihood values obtained by RAxML for those runs were -44149.18 (150_SC) and -77189.78 (150_ARB) respectively.

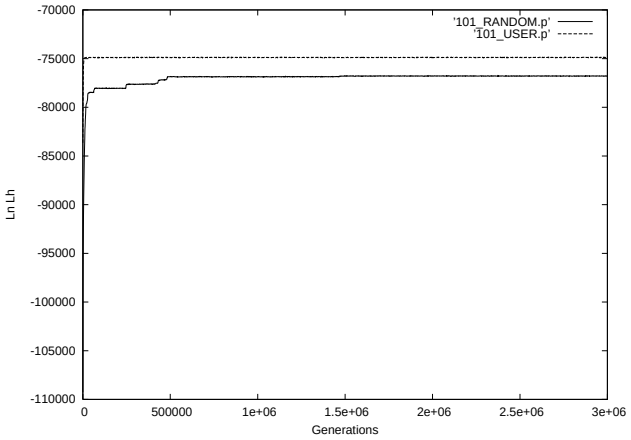


**Figure 8. Convergence behaviour of MrBayes for 500_ARB with user and random starting trees**

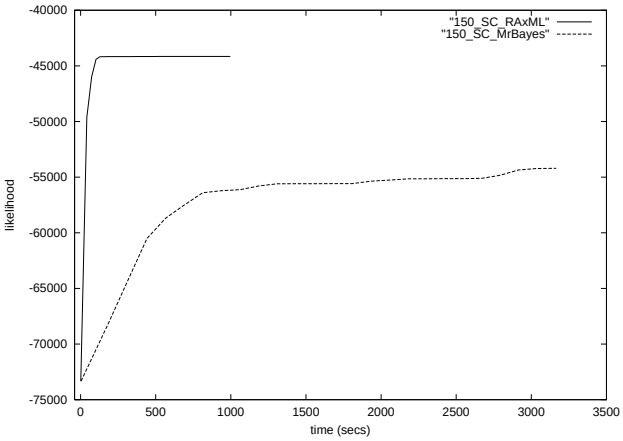


**Figure 9. 150_SC likelihood improvement over time of RAxML and MrBayes for the same random starting tree**

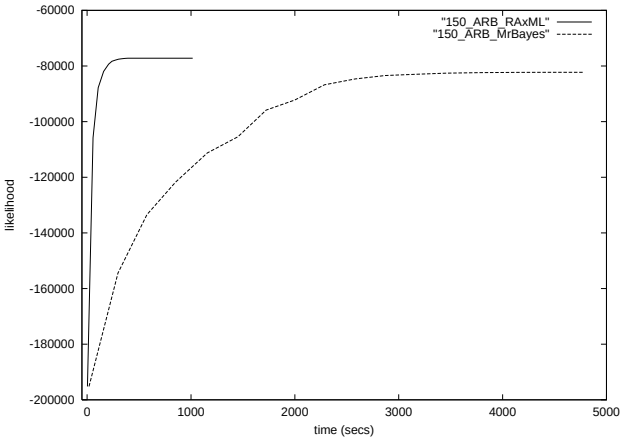


**Figure 10. 150_ARB likelihood improvement over time of RAxML and MrBayes for the same random starting tree**

**Processor Architecture:** Since our laboratory traditionally focuses on hardware architectures we used RAxML among several other applications to benchmark various PC processors. We executed RAxML with 150_SC for the same restart tree on the CPUs which are listed in Table 3 together with the respective compilers. The compiler optimization flag was set to -O3. We evaluated the effect of several advanced compiler options such as e.g. loop unrolling, frame pointer omission or higher -O values but did not observe any notable improvement of execution times. The most interesting result is that RAxML executes best on the AMD processor although the program has only been compiled with gcc.

| CPU | compiler | secs |
|---|---|---|
| AMD Opteron 244 | gcc-3.3.1 | 335 |
| Intel Xeon 2.4 GHz | gcc-3.3.1 | 559 |
| Intel Xeon 2.4 GHz | icc-7.1 | 465 |
| Intel Itanium 1.3 GHz (64bit code) | icc-7.1 | 512 |

**Table 3. RAxML execution times on PC processors**

**Distributed & Parallel Tests:** In order to test our distributed and parallel prototypes we executed our code for a fixed starting tree on 4, 8, and 16 processors of the 82-node Xeon 2.66GHz Linux cluster at the RRZE [12] with the 1000_ARB alignment. Due to the design of the distributed implementation as described in Section 4 we can not expect near-optimal speedups and exactly equivalent results for runs with distinct numbers of processors. The sequential execution required 53002 seconds and yielded a final likelihood value of -400970.31 (this is not the run which produced the best result). The times at which the distributed and parallel program passed the likelihood of the sequential one as well as the final likelihood values of the distributed code and speedups are listed in Table 4. For calculating the speedup we do not count the one processor on which the master process runs, since it does hardly produce any CPU load. The slightly superlinear speedups obtained by the parallel code are achieved due to the non-determinism of the algorithm.

| # procs. | likelihood | secs | dist. speedup | par. speedup |
|---|---|---|---|---|
| 1 (1) | -400970.31 | 53002 | 1 | 1 |
| 4 (3) | -400945.43 | 17871 | 2.97 | 2.19 |
| 8 (7) | -400950.58 | 10693 | 4.96 | 8.18 |
| 16 (15) | -400947.24 | 7542 | 7.03 | 15.31 |

**Table 4. Distributed performance of RAxML**

## 6. Conclusion, Current & Future Work

We have presented very simple heuristics for phylogenetic tree inference which outperform the currently, to the best of our knowledge, fastest and most accurate programs for phylogenetic tree inference for real-world data. Tree inference for synthetic data sets using RAxML is equally accurate as with PHYML. MrBayes has shown to be slightly more accurate for synthetic data than RAxML and PHYML but is significantly slower. The sequential code of RAxML including all test data sets and final tree topologies is freely available at: `wwwbode.cs.tum.edu/~stamatak`. In

addition we present a distributed version of our algorithm which enables seti@home-like computations of large phylogenetic trees due to its low communication costs. Furthermore, 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. To conclude we want to stress two major points within that context: *Firstly*, it seems to be important to assess performance of programs for phylogenetic tree inference with synthetic as well as real data sets, since the differences in program behaviour became apparent only with real data sets in our experiments. *Secondly*, we believe that bayesian and traditional approaches to phylogenetic inference should be combined in experimental practice, in order to exploit the advantages of both approaches.

Future work will mainly cover the execution of large parallel and distributed production runs in order to explore the maximum size of trees which can be computed using RAxML.

Finally, we plan to integrate our dnapars component as well as the fast tree optimization function into PHYML. We believe that this will improve PHYML since it lacks an option to generate distinct starting trees and perform more exhaustive optimizations. Moreover, in contrast to PHYML the RAxML optimization process is relatively straight-forward to parallelize. PHYML on the other hand offers a larger variety of evolutionary models, is able to handle protein data and includes some advanced features for model parameter optimization.

## 7. Acknowledgements

We would especially like to thank Stephane Guindon for his useful comments which greatly helped to improve the final manuscript, as well as for providing us the PHYML source code. Furthermore, we would like to thank the RRZE and HeLiCS HPC-teams for providing us access and support for their PC clusters. Finally, we are grateful to Martin Mairandres and Tobias Klug from the Lehrstuhl für Rechnertechnik und Rechnerorganisation for providing us access to our local PC cluster and testing RAxML with various processor/compiler combinations.

## References

[1] J. Felsenstein. Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *J. Mol. Evol.*, 17:368–376, 1981.

[2] S. Guindon and O. Gascuel. A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Syst. Biol.*, 52(5):696–704, 2003.

[3] HeLiCS: HEidelberg Linux Cluster System. HELICS.UNI-HD.DE, visited Jul 2003.

[4] M.T. Holder and P.O. Lewis. Phylogeny Estimation: Traditional and Bayesian Approaches. *Nature Reviews Genetics*, 4:275–284, 2003.

[5] J.P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 17(8):754-5, 2001.

[6] J.P. Huelsenbeck, B. Larget, R.E. Miller, and F. Ronquist. Potential Applications and Pitfalls of Bayesian Inference of Phylogeny. *Syst. Biol.*, 51(5):673–688, 2002.

[7] W. Ludwig et al. ARB: A Software Environment for Sequence Data. *Nucl. Acids Res.*, in press, 2003.

[8] G. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek. fastdnaml: A Tool for Construction of Phylogenetic Trees of DNA Sequences using Maximum Likelihood. *Comput. Appl. Biosci.*, 10:41–48, 1994.

[9] PAML Manual (Information on Tr/Tv definitions: page 20) BCR.MUSC.EDU/MANUALS/PAMLDOC.PDF, visited Nov 2003.

[10] PAUP project site. PAUP.CSIT.FSU.EDU, visited May 2003.

[11] PHYLIP downlaod site and list of phylogeny software. EVOLUTION.GENETICS.WASHINGTON.EDU, visited Nov 2003.

[12] Regionales Rechenzentrum Erlangen: HPC services. WWW.RRZE.UNI-ERLANGEN.DE, visited Oct 2003.

[13] Seti@home project site. SETIATHOME.SSL.BERKELEY.EDU, visited Jul 2003.

[14] A. P. Stamatakis, T. Ludwig, and H. Meier. A Fast Program for Maximum Likelihood-based Inference of Large Phylogenetic Trees In *Proceedings of ACM Symposium on Applied Computing '04*, to be published.

[15] A. P. Stamatakis, T. Ludwig, and H. Meier. RAxML: A Parallel Program for Phlogenetic Tree Inference. Poster abstract in *Proceedings of ECCB2003*, September 2003.

[16] A. P. Stamatakis, T. Ludwig, H. Meier, and M. J. Wolf. Accelerating Parallel Maximum Likelihood-based Phylogenetic Tree Computations using Subtree Equality Vectors. In *Proceedings of SC2002*, November 2002.

[17] C. Stewart, D. Hart, D. Berry, G. Olsen, E. Wernert, and W. Fischer. Parallel Implementation and Performance of fastdnaml - a Program for Maximum Likelihood Phylogenetic Inference. In *Proceedings of SC2001*, November 2001.

[18] K. Strimmer, A.v. Haeseler. Quartet Puzzling: A Maximum-Likelihood Method for Reconstructing Tree Topologies. *Mol. Biol. Evol.* 13:964-969, 1996.

[19] T.L. Williams, B.M.E. Moret. An Investigation of Phylogenetic Likelihood Methods. In *Proceedings of 3rd IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03)*, 2003.

[20] C. Tuffley, M. Steel. Links between Maximum Likelihood and Maximum Parsimony under a Simple Sodel of Site Substitution. *Bull Math Biol* 59(3):581-607, 1997.

[21] M. Wolf, S. Easteal, M. Kahn, B. McKay, and L. Jermiin. TrExML: A Maximum Likelihood Program for Extensive Tree-space Exploration. *Bioinformatics*, 16(4):383–394, 2000.