A Fast Program for Maximum Likelihood-based Inference of Large Phylogenetic Trees*

A.P. Stamatakis
Technical University of Munich
Dept. of Computer Science
Boltzmannstr. 3 D-85748
München, Germany
stamatak@cs.tum.edu

T. Ludwig
University of Heidelberg
Dept. of Computer Science
Im Neuenheimer Feld 348
D-69120 Heidelberg, Germany
t.ludwig@computer.org

H. Meier
Technical University of Munich
Dept. of Computer Science
Boltzmannstr. 3 D-85748
München, Germany
meierh@cs.tum.edu

ABSTRACT

The computation of large phylogenetic trees with maximum likelihood is computationally intensive. In previous work we have introduced and implemented algorithmic optimizations in **PAxML**. The program shows run time improvements $>$ 25% over **parallel fastDNAml** yielding exactly the same results. This paper is focusing on computations of large phylogenetic trees ($>$ 100 organisms) with maximum likelihood. We propose a novel, partially randomized algorithm and new parsimony-based rearrangement heuristics, which are implemented in a sequential and parallel program called **RAxML**.

We provide experimental results for real biological data containing 101 up to 1000 sequences and simulated data containing 150 to 500 sequences, which show run time improvements of factor 8 up to 31 over **PAxML** yielding equally good trees in terms of likelihood values and RF distance rates at the same time. Finally, we compare the performance of the sequential version of **RAxML** with a greater variety of available ML codes such as **fastDNAml, AxML** and **MrBayes**. **RAxML** is a freely available open source program.

Keywords

Phylogenetic Inference, Maximum Likelihood, Parallel & Distributed Computing

*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 HEidelberg LInux Cluster System (HELICS).

SAC'04, March 14-17, 2004, Nicosia, Cyprus

1. INTRODUCTION

In recent years there has been an astonishing accumulation of genetic information for many different organisms. This information can be used to infer evolutionary relationships (called a *phylogenetic tree* or *phylogeny*) among a collection of species. There are a variety of techniques that are used to compute these relationships, including the use of maximum likelihood which is considered to be one of the most sound methods. The number of possible tree topologies grows exponentially with the number of taxa and the computational cost of the likelihood function itself is high. Thus, the introduction of heuristics for reducing the search space in terms of potential tree topologies evaluated becomes inevitable for the computation of trees containing more than 15 to 20 organisms. Heuristics for maximum likelihood-based phylogenetic tree calculations still remain computationally intensive, mainly due to the high cost of the tree likelihood function, which is invoked repeatedly for each analyzed tree topology.

Thus, only relatively small trees of high quality (150 [17], 228 taxa with genetic algorithms, see Section 2) can be calculated so far, although large data sets containing potential phylogenetic information are available (e.g. approximately 30.000 sequences in the ARB [8] ssu rRNA database).

In previous work [15] we have introduced Subtree Equality Vectors (SEVs) to significantly accelerate the topology evaluation function of maximum likelihood-based phylogeny programs. We implemented SEVs in **PAxML** (Parallel A(x)ccelerated Maximum Likelihood), which was derived from **parallel fastDNAml** [17]. **PAxML** shows run time improvements of approximately 25% to 65% compared to **parallel fastDNAml**, yielding best accelerations for large alignments ($\geq$ 150 sequences) on PC processor architectures. The algorithmic optimizations of **PAxML** focus on obtaining *exactly* the same result as **parallel fastDNAml** in less time. The goal of current work on **RAxML** (Randomized A(x)ccelerated Maximum Likelihood) is to obtain equally good likelihood values as **PAxML** in less time using a novel algorithm. Furthermore, the new algorithm enables the implementation of a more coarse grained parallelism than comparable phylogeny programs in order to facilitate the development of its distributed version: **RAxML@home**.

2. RELATED WORK

As already mentioned heuristics, such as e.g. the stepwise addition algorithm (introduced in [3], modified in [9]), the advanced stepwise addition algorithm [21] or quartet puzzling [18], are required for the tree building process. Quartet puzzling yields trees of comparable quality but is slower than stepwise addition and thus not well-suited for reconstruction of big trees. One of the main shortfalls of the stepwise addition algorithm as implemented in **fastDNAml** and **AxML** is that the final tree strongly depends on the input order of the sequences. Thus, it is recommended to run the program several times with different randomized sequence input order permutations (jumbling) at high rearrangement levels to obtain reliable results. However, this practice becomes prohibitive for large trees ($>$ 200 sequences) even for parallel implementations of the above programs on supercomputers since thorough rearrangements and an augmentation in the number of taxa and jumbles increases run time by orders of magnitude. Parallel MPI-based implementations also exist for most sequential programs mentioned above.

As alternative to the above approaches implementations of genetic algorithms for maximum likelihood-based phylogenetic tree inference have been proposed e.g. in [1]. We have however not been able to obtain the code of those programs for conducting a comparative analysis with **PAxML** or **RAxML**. The performance of genetic algorithms has mainly been assessed using **PAUP** [14] for calculating reference trees and reference run-times.

Finally, there exist bayesian approaches to phylogenetic inference which have been implemented e.g. in **MrBayes** [6]. However, bayesian approaches require good starting and reference trees to accelerate computations and to determine when the Markov Chain Monte Carlo (MCMC) process has converged (see Section 4). These starting trees can quickly be obtained using **RAxML**.

For a good comparative analysis of phylogenetic tree building methods see [20].

3. A RANDOMIZED APPROACH

As already mentioned the impact of sequence input ordering on final results is one of the main shortfalls of **PAxML**. Thus, it is recommended to run **PAxML** several times with different randomized sequence input orderings (jumbling) in order to obtain reliable results. In order to handle this problem, we have conducted experiments assessing a variety of approaches to accelerate the program which are outlined in [16].

Those experiments lead to the following algorithm consisting of 3 major computational steps which we describe below including the rationale and implementation of each step.

Step 1: Calculate an initial set of parsimony trees for a specified number of randomized sequence input orderings (in all experiments we used number of species * 0.5). Thereafter, calculate the Ln likelihood values for those trees and store them in an ordered tree list tl. Let n be the number of trees in tl and t_1,...,t_n the topologies stored in tl.

Rationale 1: In order to accelerate the inference of randomized input order permutations we used **dnapars** which implements a similar stepwise addition algorithm as **PAxML**. We exploit the relationship between parsimony and maximum likelihood methods described e.g. in [2][19] to obtain at least equally good trees (in terms of likelihood values) compared to the initial version of **RAxML** which applied maximum likelihood-based stepwise addition without rearrangements for computing the initial tree set tl and thus required more time, without yielding better trees.

Implementation 1: For calculating parsimony trees we integrated **dnapars** from PHYLIP [12] into **RAxML**. Furthermore, we used the evaluation function from **AxML** to calculate likelihood values for the trees produced by **dnapars**.

Step 2: Calculate a majority-rule consensus tree and likelihood values for all subsets $\{t_1, t_2\}$, $\{t_1, t_2, t_3\}$, ..., $\{t_1, ..., t_n\}$ of tl. Insert the resulting consensus trees into tl.

Rationale 2: In many experiments one or several consensus trees showed a better likelihood than the best tree obtained during **Step 1** of **RAxML**. Furthermore, the consensus trees provide valuable information about frequently appearing subtrees, which can be used for a further refinement of the rearrangement process of **Step 3**.

Implementation 2: We integrated **consense** [7] from PHYLIP into **RAxML**.

Step 3: Optimize the best tree of tl by applying local and regional rearrangements in a similar way as **PAxML** but using parsimony-based heuristics. However, all rearranged topologies are initially scored by parsimony and only a fraction of trees (fraction dependent on the rearrangement setting) with the best parsimony scores is evaluated with maximum likelihood. While no better tree is found the fraction is increased progressively until all rearranged topologies have been evaluated.

Rationale 3: Especially with real biological data the best tree of **Step 1** and **Step 2** has a significantly worse likelihood than the best-known tree. Thus, rearranging the best tree becomes inevitable for further improving its likelihood score. Since the rearrangement process is the most computationally intensive part of **RAxML** (e.g. 371.00 secs at **Step 1 & 2** and 3359.34 secs at **Step 3** for 101_SC, see Section 4) we decided to exploit once again the relationship between maximum likelihood and parsimony to accelerate the rearrangement phase. The acceleration is achieved due to the significantly lower cost of the parsimony function and the very frequent appearance of improved trees among topologies with good parsimony scores. E.g. an evaluation of all rearranged topologies for 150_ARB (see Section 4) with parsimony requires 47.92 secs whereas an evaluation with maximum likelihood takes 7023.75 secs.

Implementation 3: For reasons of efficiency we implemented the parsimony-score function directly in **RAxML**.

The parallel version of **RAxML** communicates via MPI and consists of a simple master-worker architecture. It is available for download as open source code at [11].

4. TEST DATA & RESULTS

For our experiments we extracted alignments comprising 150, 200, 250, 500 and 1000 taxa (150_ARB,...,1000_ARB) from the ARB [8] small subunit ribosomal ribonucleic acid (ssu rRNA) database containing organisms from the kingdoms 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 [10]. Finally, we generated

several simulated trees and respective alignments containing 150, 200, 250 and 500 sequences (150_SIM,...,500_SIM) using the HKY (Hasegawa et al. 1985) model of sequence evolution, various combinations of base frequencies, transition/transversion ratios as well as different models of rate heterogeneity.

Parallel Tests:

All tests have been conducted on the HELICS [5] 512 processor Linux Cluster using 32 up to 200 processors for the largest alignments.

In Table 1 we summarize the results for our experiments with real biological data. For each data set we use the best-known tree in terms of Ln likelihood values as reference tree and provide the total amount of CPU hours required by **PAxML** and **RAxML**. Furthermore, for each run we indicate the final likelihood value and the Robinson-Foulds (RF) rates indicating the topological distance to the reference tree. Note that the values and topologies of the best-known trees for all data sets, except the 250_ARB alignment, have been inferred with program versions of **RAxML**. **RAxML** is particularly well suited for large data sets such as the 1000_ARB alignment since it constantly returns trees with a significantly better final likelihood than **PAxML** in less time. The inference of the best-known reference tree for 1000_ARB with higher rearrangement parameters and a greater number of random input order permutations required 4114.69 CPU hours on HELICS, which is still faster by factor 2.41 than **PAxML** and at least 4.82 than **parallel fastDNAml** (see [15]).

In Table 2 we outline the results obtained for simulated data respectively. Note that in those cases were **RAxML** did not return the model tree, it was contained in the tree file list *tl* the program returns.

We consider the results obtained with real biological data containing errors and gaps more meaningful although the "real" tree is not known since we observed that both **PAxML** and **RAxML** converged much faster and constantly to the model tree for simulated data. Thus, we believe that inferring trees for real error-prone data is much harder than for simulated alignments and maximum likelihood programs should also be evaluated based upon a standard benchmark set of real alignments.

Sequential Tests:

All sequential tests were performed on an Intel Xeon 2.2 GHz Processor. We compiled the programs using `icc -O3` (native Intel compiler).

In Table 3 we list execution times, final likelihood values and RF distances to the best-known tree of **AxML, fastDNAml, RAxML** and **MrBayes** for the 101_SC data set from which we removed the rate category vector to speed up computations. For these initial tests we used the HKY (Hasegawa et al. 1985) model of sequence evolution and uniform rates among sites. The rearrangement Level of **AxML, fastDNAml, RAxML** was set to 5. **MrBayes** was run twice with 4 chains using a random starting tree and a user starting tree. The user starting tree was computed using **RAxML** within 365 seconds and has a likelihood of -74091.15.

In Figure 1 we plot the likelihood values of a SC_101 optimization process over time for **RAxML** and **MrBayes** with the same user starting tree. The execution time for **MrBayes** with the user starting tree is the time of the first appearance of the best likelihood value whereas the **MrBayes** run with a random starting tree was aborted at the specified execution time.

In this experiment **RAxML** clearly outperforms all other programs in terms of execution time and final likelihood value except for **MrBayes** with the user starting tree.

As can be derived from Table 3 and Figures 5 and 7 however **MrBayes** requires significantly longer execution times for de novo tree computations, since it does not provide functionality for computing good starting trees and uses a random tree as initial tree.

Table 3: Execution times (secs), Ln likelihoods, RF rates for fastDNAml, AxML, RAxML, MrBayes

Program	Seconds	Ln likelihood	RF rate
AxML	118534.60	-73975.90	13.07%
fastDNAml	179745.32	-73975.90	13.07%
MrBayes (User)	1253.11	-73962.63	0.0%
MrBayes (Random)	25908.23	-76676.27	26.13%
RAxML	3742.26	-73962.63	0.0%

In Figure 2 we display the improvement of the Likelihood value (HKY model of sequence evolution) over time for the sequential programs **fastDNAml, AxML** and **RAxML** during the optimization of a final tree for ARB_150 with an initial Ln likelihood of -77258.24. Figure 2 shows the impact of the parsimony-based rearrangement heuristics on tree optimization time. Furthermore, in Figure 3 we compare **RAxML** with **MrBayes** for the same optimization process. **MrBayes** and **RAxML** perform equally well in tree optimization provided the same starting tree.

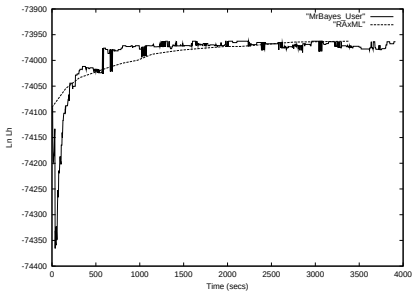

Figure 1: Likelihood improvement over time for SC_101 optimization

Since it appears that running **MrBayes** with "good" user trees can speed up convergence and improve final results we performed some additional tests with the SC_101 and SC_150 data sets under the general reversible model of sequence evolution. The starting tree for SC_150 could be computed within 1729 seconds with **RAxML**. We executed **MrBayes** with 4 chains and let it run for 3.000.000 generations. We depict the development of the Ln likelihood value over the number of generations for SC_101 for the first 100.000 and 3.000.000 generations in Figures 4 and 5 respectively. In Figures 6 and 7 we plot the same information for data set SC_150 respectively. For 3.000.000 generations **MrBayes** required 54233 (SC_101) and 67013 (SC_150) seconds respectively. Those initial tests show that "good" starting

Table 1: CPU hours, Ln likelihoods, RF rates for PAxML/RAxML (real data)

Data	best-known tree Ln Lh	CPU hrs PAxML	Ln Lh	RF rate	CPU hrs RAxML	Ln Lh	RF rate	acc
150_SC	-44145.98	163.66	-44146.90	9.43%	20.43	-44145.98	0.00%	8.01
150_ARB	-77189.69	300.40	-77189.78	2.70%	19.70	-77193.73	2.02%	15.25
200_ARB	-104743.33	774.56	-104743.33	0.00%	41.71	-104743.33	0.00%	18.57
250_ARB	-131468.97	1947.18	-131468.97	0.00%	80.69	-131479.99	2.80%	24.13
500_ARB	-252553.12	7371.79	-252588.67	8.42%	889.43	-252553.12	0.00%	8.29
1000_ARB	-401242.80	9898.05	-402282.08	13.37%	1070.59	-401501.57	6.21%	9.23

Table 2: CPU hours, Ln likelihoods, RF rates for PAxML/RAxML (simulated data)

Data	simulated tree Ln Lh	CPU hrs PAxML	Ln Lh	RF rate	CPU hrs RAxML	Ln Lh	RF rate	acc
150_SIM	-49401.09	133.00	-49398.73	1.02%	5.03	-49398.73	1.02%	26.44
200_SIM	-68418.88	281.08	-68418.88	0.00%	17.08	-68418.88	0.00%	16.46
200_2_SIM	-96158.55	445.21	-96158.55	0.00%	25.99	-96158.55	0.00%	17.13
250_SIM	-86348.99	531.63	-86348.99	0.00%	17.33	-86348.99	0.00%	30.67
500_SIM	-166185.70	1670.06	-166185.33	0.10%	178.25	-166185.33	0.10%	9.37

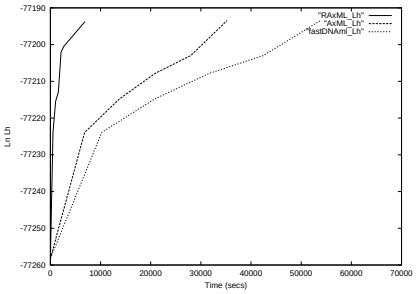

Figure 2: Likelihood improvement over time for ARB_150 optimization

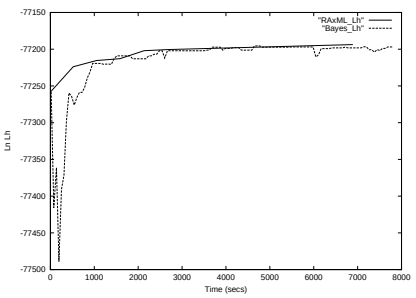

Figure 3: Likelihood improvement over time for ARB_150 optimization

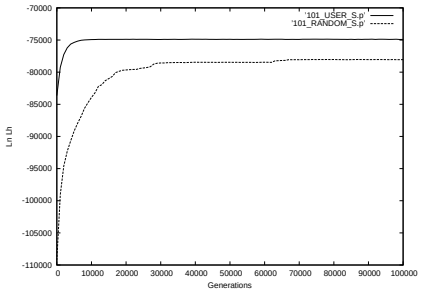

Figure 4: Likelihood improvement over 100.000 generations for SC_101

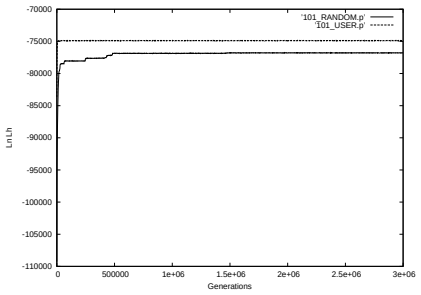

Figure 5: Likelihood improvement over 3.000.000 generations for SC_101

trees lead to significantly better likelihood values and reduce the number of required generations for convergence. Furthermore, starting trees seem to induce a smoother convergence to a stationary state and might be of great help since deciding when to stop an MCMC simulation is the "$64,000 question for MCMC analysis" as Huelsenbeck puts it in the **MrBayes** manual. Thus, we recommend using **RAxML** to compute good starting trees for **MrBayes** and to obtain an estimate at which likelihood value the Markov chain of **MrBayes** should be interrupted which is the major problem with bayesian approaches. We have already integrated an additional function into the new version of **RAxML** (see Section 5 below) which writes a complete **MrBayes** input block with a good starting tree obtained by **RAxML** to a file, in order to avoid tedious file conversions to NEXUS format.

5. CURRENT & FUTURE WORK

Currently, we are developping the new version of **RAxML** which incoorporates a slightly different approach from the program version presented in this paper. Our program starts with one single parsimony tree and implements an altered rearrangement process which enables simultaneous application of topological changes within one rearrangement cycle.

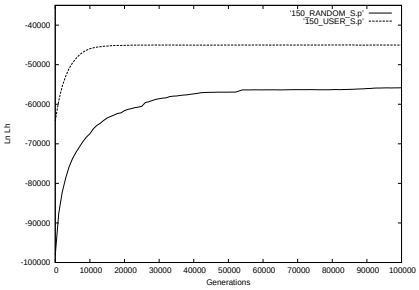

Figure 6: Likelihood improvement over 100.000 generations for SC_150

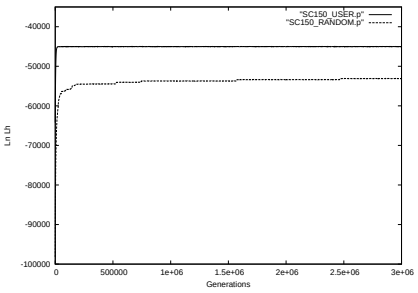

Figure 7: Likelihood improvement over 3.000.000 generations for SC_150

Furthermore, due to a reduced application of branch length optimization operations which have a less significant effect on the overall likelihood than the topology per se the average evaluation time per topology has been reduced by orders of magnitude. The new program enables computation of trees containing up to 1000 taxa within less than 24 hours on a single Intel Xeon 2.4GHz CPU (see Table 4 for some results). Note, that especially for large trees signifcantly better results were obtained.

Table 4: Results for the new version of RAxML

Data	Seconds (Hours:Minutes)	Ln likelihood
101_SC	616.7 (00:10)	-73919.30
150_SC	389.5 (00:07)	-44142.61
150_ARB	178.0 (00:03)	-77189.70
200_ARB	271.5 (00:05)	-104742.56
250_ARB	1067.5 (00:18)	-131468.02
500_ARB	26123.7 (07:12)	-252499.43
1000_ARB	50729.1 (14:06)	-400925.30

Future work will cover the implementation of a parallel MPI-based program, as well as the integration of the new algorithm into **RAxML@home**.

6. REFERENCES

[1] M.J. Brauer et al. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. In *Molecular Biology and Evolution* 19:1717-1726, 2002.

[2] R.W. DeBry, L. G. Abele. The relationship between parsimony and maximum likelihood analyses: tree scores and confidence estimates. In *Mol. Biol. Evol.* 12:291-297, 1995.

[3] J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. In *J. Mol. Evol.* 17:368-376, 1981.

[4] P.A. Goloboff. Analyzing Large Data Sets in Reasonable Times: Solutions for Composite Optima. In *Cladistics* 15:415-428, 1999.

[5] HEidelberg LInux Cluster System: `helics.uni-hd.de`. Technical Information about the HELICS cluster.

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

[7] L.S. Jermiin et al. Majority-rule consensus of phylogenetic trees obtained by maximum-likelihood analysis. *Mol. Biol. Evol.* 14:1297-1302, 1997.

[8] W. Ludwig et al. ARB: a software environment for sequence data. In *Nucl. Acids Res.*, in press, 2003.

[9] G.J. Olsen et al. fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. In *Comput. Appl. Biosci.* 10:41-48, 1994.

[10] **parallel fastDNAml** `www.indiana.edu/~rac/hpc/fastDNAml`. Download site including additional information results and test data.

[11] **PAxML/RAxML** download site. `wwwbode.in.tum.de/~stamatak/research.html`

[12] PHYLIP `evolution.genetics.washington.edu/phylip.html`. Downlaod site for PHYLIP version 3.5, version 3.6 alpha and related documentation.

[13] **RAxML@home** development site. `www.sourceforge.com/projects/axml`

[14] **PAUP** download and information site. `paup.csit.fsu.edu`

[15] A.P.Stamatakis et al. Accelerating Parallel Maximum Likelihood-based Phylogenetic Tree Computations using Subtree Equality Vectors. In *Proceedings of Supercomputing Conference (SC2002)*, Baltimore, Maryland, November 2002.

[16] A.P.Stamatakis, T. Ludwig. Phylogenetic Tree Inference on PC Architectures with AxML/PAxML. In *Proceedings of IPDPS2003, High Performance Computational Biology Workshop (HICOMB)*, Nice, France, April 2003.

[17] C. A. Stewart et al. Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference. In *proceedings of SC2001*, Denver, Colorado, November 2001.

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

[19] C. Tuffley, M. Steel. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. In *B*ull Math Biol 59(3):581-607, 1997.

[20] 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.

[21] M.J. Wolf et al. TrExML: A maximum likelihood program for extensive tree-space exploration. In *Bioinformatics* 16:383-394, 2000.