An Efficient Program for Phylogenetic Inference Using Simulated Annealing[*]

Alexandros Stamatakis
Institute of Computer Science, Foundation for Research and Technology-Hellas
P.O. Box 1385, Heraklion, Crete, GR-71110 Greece
stamatak@ics.forth.gr

Abstract

Inference of phylogenetic trees comprising thousands of organisms based on the maximum likelihood method is computationally expensive. A new program RAxML-SA (Randomized Axelerated Maximum Likelihood with Simulated Annealing) is presented that combines simulated annealing and hill-climbing techniques to improve the quality of final trees. In addition, to the ability to perform backward steps and potentially escape local maxima provided by simulated annealing, a large number of "good" alternative topologies is generated which can be used to build a consensus tree on the fly. Though, slower than some of the fastest hill-climbing programs such as RAxML-III and PHYML, RAxML-SA finds better trees for large real data alignments containing more than 250 sequences. Furthermore, the performance on 40 simulated 500-taxon alignments is reasonable in comparison to PHYML. Finally, a straight-forward and efficient OpenMP parallelization of RAxML is presented.

1. Introduction

Phylogenetic trees are used to represent the evolutionary history of a set of n organisms which are often also called taxa within this context. A multiple alignment of a small region of their DNA or protein sequences can be used as input for the computation of phylogenetic trees. Note, that a high-quality multiple alignment of the organisms is a necessary prerequisite to conduct a phylogenetic analysis: *The quality of the evolutionary tree can only be as good as the quality of the multiple alignment!*

In a computational context phylogenetic trees are usually strictly bifurcating unrooted trees. The organisms of the alignment are located at the tips of such a tree and the inner nodes represent extinct common ancestors. The branches of the tree represent the time which was required for the mutation of one species into another—new—one. The inference of phylogenies with computational methods has many important applications in medical and biological research, such as e.g. drug discovery and conservation biology (see [1] for a summary). Due to the rapid growth of available sequence data over the last years and the constant improvement of multiple alignment methods it has now become feasible to compute very large trees which comprise more than 1.000 organisms. The computation of the tree-of-life containing representatives of all living beings on earth is considered to be one of the *grand challenges* in Bioinformatics.

The most fundamental algorithmic problem computational phylogeny faces consists in the immense amount of potential alternative tree topologies. This number grows exponentially with the number of sequences n, e.g. for $n = 50$ organisms there already exist $2.84 * 10^{76}$ alternative topologies; a number almost as large as the number of atoms in the universe ($\approx 10^{80}$). Thus, given some—biologically meaningful—optimality criterion for evaluating all alternative configurations in order to search for the best tree, one can quickly assume that the problem might be NP-hard. In fact, this has already been demonstrated for the *maximum parsimony* (MP) criterion [5]. The *maximum likelihood* (ML) criterion [6] is also believed to be NP-hard, though this could not be demonstrated so far due to the mathematical complexity of the model.

Another important aspect for the design of heuristic tree searches consists in the very high degree of accuracy (difference to the score of the optimal or best-known solution) which is required to obtain reasonable biological and topologically closely related results. While an accuracy of 90% is considered to be a "good" value for heuristics designed to solve other NP-hard optimization problems, recent results suggest [30] that phylogenetic analyses require an accuracy $\geq$ 99.99%. When comparing the various optimality criteria for phylogenetic trees one can observe a *trade-off* between speed and quality. This means that a phylogenetic analysis conducted with an elaborate model, e.g. ML requires significantly more time but yields trees with superior accuracy

* This work is supported by a Postdoc-fellowship granted by the German Academic Exchange Service (DAAD).

than e.g. *neighbor joining* [7] (NJ) or MP [8, 29]. Due to the higher accuracy it is desirable to infer large and complex trees with compute-intensive statistical methods. It is important to emphasize that the design of maximum likelihood programs is primarily an *algorithmic discipline*, due to the gigantesque number of alternative tree topologies. Thus, progress in the field is mainly attained via algorithmic improvements rather than by brute force allocation of computational resources. Therefore, the main focus of this paper is on algorithmic as well as technical solutions for the computation of large trees (containing $\geq$ 500 sequences) based on statistic models of sequence evolution. Moreover, the intention is to combine the advantages of hill-climbing and simulated annealing searches in order to produce more reliable results.

The remainder of this paper is organized as follows: In Section 2 related work is briefly described with a focus on current state-of-the-art sequential programs for maximum likelihood-based inference, which are used to assess performance of RAxML-SA. In the following Section 3 the simulated annealing algorithm is outlined. Section 4 summarizes experimental results for RAxML-SA on simulated and real data. The subsequent Section 5 describes the OpenMP parallelization of RAxML. Finally, Section 6 provides a conclusion and addresses current and future issues of work.

2. Related Work

The survey of related work is restrained to statistical phylogeny 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 [19] 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 [11]. In fact, Bayesian methods also use a search technique based on simulated annealing with multiple chains which is known as Metropolis-Coupled Markov-Chain Monte-Carlo (MC^3) algorithm. A thorough comparison of popular phylogeny programs using statistical approaches such as fastDNAml, MrBayes, PAUP [18], and TREE-PUZZLE [26] on *small* simulated datasets (up to 60 sequences) has been conducted by T.L. Williams *et al* [29]. The most important result of this paper is that MrBayes outperforms all other phylogeny programs in terms of speed and tree quality. However, the results of this survey do not necessarily apply to large real data sets since simulated alignment data has different properties and a significantly stronger phylogenetic signal than real world data (see Section 4 for a discussion), i.e. typically much more computational effort is required to find a "good" phylogenetic tree for real-world data. Due to the significant differences between real and simulated datasets comparative surveys should include collections of simulated *and* real datasets in order to yield a more complete image of program performance. In fact, there exist some real datasets for which MrBayes fails to converge to acceptable likelihood values within reasonable time [25]. Huelsenbeck *et al* [12] provide an in-depth discussion of potential pitfalls of Bayesian inference. More recently, Guidon and Gascuel published an interesting paper about their new program PHYML [8], which is very fast and seems to be able to compete with MrBayes. PHYML is a "traditional" maximum likelihood hill-climbing program which seeks to find the optimal tree in respect to the likelihood value and like MrBayes is also capable of optimizing nucleotide substitution model parameters. Moreover, the respective performance analysis includes larger simulated datasets of 100 sequences and two well-studied real data sets containing 218 and 500 sequences which are also used in the current paper. Their experiments show that PHYML is extremely fast on real and simulated data. However, the accuracy on real data needs improvement [25]. Moreover, the results show that well-established sequential programs like PAUP* [18], TREE-PUZZLE [26], and fastDNAml [16] are prohibitively slow on datasets containing more than 200 sequences, at least in sequential execution mode. Therefore, PAUP*, TREE-PUZZLE, and fastDNAml are not included in the present study (see [8] for performance data of these programs). More recently Vinh *et al* [27] published a program called IQPNNI which yields better trees than PHYML on real world data but is significantly slower. It is important to note that the authors of IQPNNI also included a special program option to distinguish between inference of real and simulated data alignments due to their distinct characteristics. Finally, the current hill-climbing version of RAxML-III clearly outperforms PHYML on real world data, both in terms of execution time and final tree quality [25]. However, it constantly performs worse than PHYML on simulated data due to the aforementioned properties of simulated data and the points discussed in Section 4. The first application of simulated annealing techniques to ML tree searches is proposed by Salter *et al* [22] (the technique has previously been applied to MP phylogenetic tree searches by Barker *et al* [2]). However, the respective program SSA has not become very popular due to the limited availability of nucleotide substitution models and its focus on the molecular clock model of evolution. Moreover, the program is relatively hard to use and comparatively slow in respect to recent hill-climbing implementations. Despite the fact that Salter *et al* were the first to apply simulated annealing to ML-based phylogenetic tree searches the author is not aware of any biological results published based on SSA. Due to the aforementioned problems and limitations of SSA it was not feasible to use the program in the comparative analysis on large datasets with PHYML and RAxML-SA. In the final analysis it can be stated that MrBayes, PAUP*, TREE-PUZZLE, and fastDNAml are too slow to conduct a comprehensive performance study on large simulated and real world data. Therefore, mainly PHYML, RAxML-III, and in some cases IQPNNI have been used to assess performance of RAxML-SA. This represents a balanced choice since PHYML performs well on simulated data and RAxML-III on real world data. The performance of IQPNNI is situated somewhere between PHYML and RAxML-III.

3. The Simulated Annealing Algorithm

The application of the simulated annealing technique to combinatorial optimization problems which are based on an optimality criterion $f()$ is introduced by various authors, notably Cerny [3] and Kirkpatrick *et al* [14]. A more detailed description of the application of simulated annealing to phylogenetic tree searches can be found in [2] and [22].

The main components of a simulated annealing algorithm are listed below:

1. A method for generating a candidate solution t_{i+1} based on the current solution t_i (often also called configuration); in the concrete case of phylogenetic analysis this methods is called topology proposal mechanism. The initial configuration t_0 is often chosen at random.
2. A problem-specific cooling schedule which determines how frequently and to which extent solutions which decrease the value of the objective function $f()$, i.e. induce backward steps, will be accepted and how this acceptance parameters are modified as the computation advances. Occasional acceptance of backward steps can help escape local maxima.
3. The Metropolis-step in which the algorithm decides if a solution t_{i+1} is accepted or rejected. If t_{i+1} improves the value of the objective function it is always accepted, otherwise it is accepted with a probability $P(f(t_i), f(t_{i+1}))$ (see below).
4. A stopping criterion, which determines when to stop the simulated annealing process.

In the implementation of RAxML-SA the stopping criterion is omitted and the program is stopped on user's discretion. However, as a rule-of-thumb RAxML-SA should be provided the four-fold execution time of its hill-climbing counterpart RAxML-III. Thus, the recommended practice for using RAxML-SA consists in initially using RAxML-III to obtain a reference execution time and likelihood value. The desired RAxML-algorithm (SA or hill climbing) can be selected by a simple command line switch. The goal of the work carried out on RAxML-SA is to integrate the efficient hill-climbing operations introduced with RAxML-III into a solid theoretical framework which allows for backward steps and avoidance of local maxima. Moreover, the computation of consensus trees on the fly is intended to yield more reliable biological results than standard hill-climbing algorithms. Thus, RAxML-SA can be regarded as a hybrid hill-climbing/simulated annealing program.

In the rest of the current Section the cooling schedule, the tree proposal mechanism, and the whole algorithm are described.

Cooling Schedule: In the concrete case the optimality criterion $f()$ corresponds to the likelihood function and the distinct tree topologies correspond to the configurations $t_0, ..., t_{i-1}, t_i, t_{i+1}, ..., t_n$.

As cooling schedule for the temperature T the simple schedule $T = T * \alpha$ is used, where $\alpha = 0.95$ and T is updated every m moves. The parameter m is set to n by default which corresponds to the number of taxa in the tree; the starting value for T is set to 3.0. In the Metropolis step a tree t_{i+1} is accepted if $P_2 \geq P_1$ where $P_1 = exp((f(t_{i+1}) - (f(t_i))/T)$ and P_2 is an equally distributed random variable between 0 and 1. Note that the signs in the formula for P_1 are correct since log likelihood values are negative. This means that if the likelihood of tree $t_{i+1}, f(t_{i+1})$ is better than that of t_i the new tree will always be accepted. In the other case t_{i+1} will be accepted sometimes; new trees t_{i+1} which have only a small difference to the score of t_i will be accepted more frequently than trees with a significantly worse likelihood score.

Tree Proposal Mechanism: The tree proposal mechanism represents the key element of the algorithm. It incorporates some important features of the RAxML-III hill-climbing algorithm, namely *lazy subtree rearrangements*. Given the current configuration (current tree) t_i the next configuration (tree) t_{i+1} is proposed either by:

1. optimizing the nucleotide substitution model parameters with probability 0.25% (**model optimization**).
2. optimizing the length of a randomly chosen branch with probability 4.75% (**branch optimization**).
3. selecting the best topology among a set of topologies generated by lazy subtree rearrangements with probability 95% (**topology proposal**).

The selection probabilities for the distinct moves represent good empirical values. The branch length optimization is carried out by a standard Newton-Raphson method and yields a tree with an at least equally good or slightly improved likelihood in respect to t_i. The same holds for the optimization of the nucleotide substitution model parameters. Thus, for model optimization and branch optimization the tree proposal mechanism performs a strict hill-climbing operation to generate t_{i+1}. This means that in case of branch or model optimization t_{i+1} is always accepted by the Metropolis-step.

Branch length optimization need not be carried out too frequently since the topology proposal step which is described below also performs a fast superficial branch length optimization. Furthermore, model optimization need not be applied frequently, since the model parameters are relatively stable over a wide range of reasonable (non-random) topologies [31]. The choice to lay out those two components as strict hill-climbers is based on the following observation: When a relatively stable state of branch length and model parameter configurations is reached the major changes in the likelihood (objective function) are caused by alterations of the topology. Thus, possible backward steps can only be triggered when the topology proposal mechanism is selected (95% of all moves represent topological alterations). The rationale for this is to provide the possibility to generate and accept potential backward moves which fit into the simulated annealing scheme while maintaining the advantages of well-established hill-climbing techniques.

As already mentioned RAxML-SA uses the concept of *lazy subtree rearrangements* to propose an alternative topology t_{i+1} which is introduced in [25]. The key idea of subtree rearrangements consists in removing a given subtree from the current tree t_i and re-inserting it into all neighboring branches with a distance of `min` up to `max` nodes from the initial deletion point of the subtree. Usually, the branch lengths of each topology generated during this process are optimized exhaustively. Lazy rearrangements do not optimize all branches of an alternative topology but only the three branches adjacent to the new insertion point, whereas the remaining branches remain unchanged. This allows for rapidly pre-scoring a large number of alternative topologies. For a more detailed description of lazy subtree rearrangements see [25].

The topology proposal mechanism in RAxML-SA works as follows: Initially, a candidate subtree for removal is selected at random. Thereafter, the minimum rearrangement distance is set to `min=1` and the maximum subtree rearrangement distance is set to `max=Random(2...rearrangementsMAX)`.

Function `Random(2...rearrangementsMAX)` returns a randomly chosen integer value between 2 and `rearrangementsMAX` which is set to 21 by default but can be adjusted by a command line switch. Given those parameters lazy subtree rearrangements of the candidate subtree are performed on the current tree t_i. After completion of all rearrangements the branch lengths of the best pre-scored rearranged topology are superficially optimized. The superficial optimization is carried out by a one-pass optimization of each individual branch in the tree. The such obtained likelihood value is then propagated to the Metropolis-step.

Finally, in order to summarize the information of the large number of "nearly" equally good trees generated during the simulated annealing process every $2n$ steps (n: number of taxa) a consensus tree is built out of the 100 best—topologically distinct—trees. The consensus tree is built by a call to the consensus tree program consense [13] which is integrated into the RAxML-SA distribution.

Whole Algorithm: In order to provide a complete description of the algorithm all major computational steps are listed below:

1. Generate a randomized parsimony starting tree (this will yield a distinct starting tree for each run) exactly as in RAxML-III *or* read in a user-specified starting tree.
2. Perform initial optimization of nucleotide substitution parameters and branch lengths.
3. Start endless simulated annealing loop.
4. Select model optimization (0.25%), branch optimization (4.75%), or topology proposal (95%).
5. Reject or accept proposed tree in the Metropolis-step.
6. Update cooling schedule parameters.
7. Every $2n$ invocations of the tree proposal mechanism build consensus tree out of 100 best and most recent trees.
8. Goto 4.

4. Results

Test Data & Platforms: For conducting experiments alignments comprising 150, 200, 250, 500, 1.000, 1.665, and 2.025 taxa (150_ARB,...,2025_ARB) have been extracted from the ARB small subunit ribosomal ribonucleic acid (ssu rRNA) database [15]. Those alignments contain organisms from the domains Eukarya, Bacteria and Archaea. In addition, the 101 and 150 sequence data sets (101_SC, 150_SC) which can be downloaded at http://www.indiana.edu/˜rac/hpc/fastDNAml are used. The 101_SC and 150_SC alignments have proved to be very hard to optimize, in terms of convergence to best-known likelihood values, especially for MrBayes (see [25]).

Furthermore, two well-known real data sets comprising 218 and 500 sequences (218_RDPII, 500_ZILLA) were included into the test set. Those two alignments are considered to be "classic" real data benchmarks and have also been used by Guindon *et al* with PHYML. In particular, the 500_ZILLA alignment has been studied extensively under the parsimony criterion [4]. A 193-taxon data set 193_V is also included which has been used by Vinh *et al* [28] to assess performance of the PhyNav program.

Finally, 40 simulated 500-taxon (α500_1,...,α500_30, π500_1,...,π500_10) alignments haven been generated using the program r8s [23]. Furthermore, Seq-Gen [20] was used as indicated below to generate the α-alignments (model incorporating heterogeneity of evolutionary rates among sites) `seq-gen -m HKY -l 1000 -t x -a y -s 0.5` with transition/transversion ratio `x` set to 1.5 *and* 2.0 for each distinct value `y` of the α shape parameter which was set to values ranging from 0.1 (high level of rate heterogeneity), 0.4, 0.7, 1.0,..., 4.3 (low level of rate heterogeneity). The parameter `-l` specifies the alignment length and `-m` the model of nucleotide substitution (HKY85 [9]). The setting for the π-alignments (plain alignments without rate heterogeneity) is outlined below: `seq-gen -m HKY -l 1000 -t x -s y` with transition/transversion ratio `x` set to 1.5 *and* 2.0 for each distinct value `x` of the branch length scaling parameter `y` which was set to values in the range of 0.5, 0.6,...,0.9.

PHYML, RAxML-III, and RAxML-SA have been compiled with the native Intel compiler `icc -O3` and executed on a cluster of unloaded Intel Xeon 2.4GHz processors equipped with 4GB of main memory for real data experiments. The simulated alignment experiments have been conducted on an unloaded Intel Centrino 1.4GHz processor with 768MB of main memory.

IQPNNI could not be re-compiled with `icc -O3` since the program is only available as LINUX-binary without source code.

Simulated Data Experiments: For the simulated data experiments one has to distinguish between the results obtained for alignments incorporating rate heterogeneity among sites (α-alignments) and those who do not (π-alignments).

α-alignments: The average execution time of PHYML for the α-alignments is 393 seconds whereas RAxML-SA was executed with a running time limitation of 3600 seconds, thus the average execution time was 1 hour. In the simulated experiments the RAxML-SA search was initiated with the BIONJ starting tree of PHYML. This is due to the fact that with simulated data the true tree need not be the maximum likelihood tree, i.e. the true tree need not have the best likelihood value. In fact, 11 out of 30 parsimony starting trees for α500_1,...,α500_30 already showed a better likelihood than the true tree. Therefore, BIONJ starting trees were used which had a better likelihood value than the true tree in only 2 out of 30 cases. This phenomenon of over-estimation is also caused by the strong and perfect phylogenetic signal produced by simulated alignment data which does not contain gaps nor alignment or sequencing errors. This perfection leads to computation of near-optimal starting trees with simpler methods such as BIONJ and MP. For example consider the relative differences in likelihood scores between starting trees and the true tree for simulated data: it is only 0.04% on average for the α and π-alignments. On the other hand it amounts to 0.76% (factor 19) for the difference between real data starting trees and best-known trees of the real data alignments used in this study. Thus, a significantly higher amount of more drastic topological changes is required to optimize real data trees which is also reflected by the significantly longer execution times for real data (see Table 1). As example for the large difference in the amount of applied topological changes between simulated and real data consider that PHYML only executed 20 NNI-swaps (Nearest Neighbor Interchange) on the α500_21 dataset whereas it executed 202 (factor 10) on the 500_ARB dataset.

The average normalized RF-distance (Robinson-Foulds distance [21]) to the true tree obtained with PHYML for the α-alignments is 0.020. For RAxML-SA 2 measures were applied: Firstly, the RF-distance to the last consensus tree written by the program which averaged to 0.023 and secondly the best RF-distance to all intermediate checkpoints written by RAxML-SA which amounts to 0.021 on average. The second measure was selected because RAxML-SA has a strong tendency to over-estimate the trees, i.e. produce trees with better likelihood values than the true tree. Thus, trees which are topologically closer to the true tree are detected during earlier stages of the RAxML-SA inference process. An example for this behavior is outlined as a plot of log likelihood values over time in Figure 1 for dataset α500_21. The likelihood value of the true tree is indicated by a thick horizontal line. The non-monotonous increase in the log likelihood score is due to the backward steps of the simulated annealing process. The average time to reach the respective best checkpoint in terms of RF-distance for RAxML-SA is 756 seconds.

π-alignments: For the 10 π-alignments PHYML yielded trees with an average normalized RF-distance of 0.0167 and required an average execution time of 98 seconds. For plain data RAxML-SA could be executed with the standard randomized parsimony starting tree and was granted an execution time limit of 1200 seconds. The average RF-distance of the last consensus tree to the true tree is 0.0166 whereas the average topological distance to the best checkpoint amounts to 0.0167. The average time required to reach the best checkpoint in terms of RF-distance is 322 seconds.

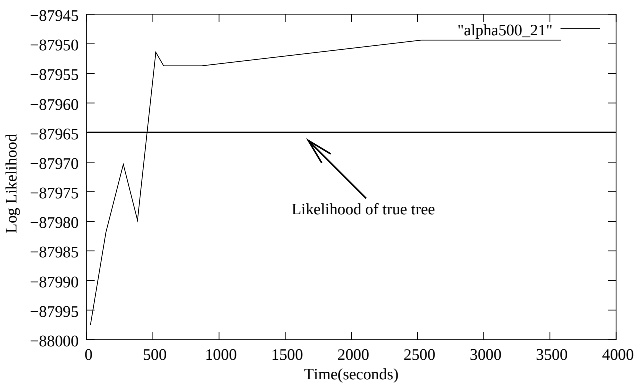

Figure 1. Over-estimation of simulated tree by RAxML-SA

The performance differences which can be observed between PHYML and RAxML-SA on real and simulated data as well as execution times are due to the distinct design of topological moves in those two programs. The modified NNI-strategy deployed in PHYML allows for local changes only whereas *lazy subtree rearrangements* in RAxML-SA allow for more drastic regional modifications of the topology. Due the significantly smaller topological and likelihood distance between starting trees and true trees for simulated data the PHYML-strategy is more efficient on this type of data. Moreover, it does not produce over-estimated trees as frequently as RAxML-SA due to the less exhaustive search. On the other hand RAxML-SA is faster and significantly better on real data as shown in the next Section.

Real Data Experiments: All tests with real data have been carried out using the HKY85 [9] model of nucleotide substitution without accounting for rate heterogeneity among sites. The transition/transversion parameter of the model was optimized by the programs. Note, that all likelihood values correspond to RAxML-likelihood values, i.e. the likelihood of the final PHYML and IQPNNI trees was evaluated with RAxML. This is due to subtle differences in the numerical implementation of the likelihood functions which mainly concern the scaling procedure for very small values. However, this has only an effect on the absolute figures and not on the relative differences among final likelihood values.

In Table 1 the results for the experiments on real alignment data sets are summarized. Column *PHYML* indicates the likelihood values of the best tree found by PHYML and the respective execution time (*secs*). The next two columns *R-III* and *secs* indicate the likelihood and execution time required for the best out of 10 (out of 5 for 2025_ARB) RAxML-III executions with distinct randomized parsimony starting trees. The next two columns (*R-III(avg), secs(avg)*) indicate the average likelihood values and execution times over those 10 runs. Columns *RAxML-SA* and *secs* contain the likelihood scores and run times of one RAxML-SA execution per dataset. Finally, the remaining columns indicate at which point of time the first checkpoint was written by RAxML-III (*R-III* $>$ *PHY*) and RAxML-SA (*R-SA* $>$ *PHY*) respectively which contained a tree with a better likelihood than the final tree yielded by PHYML. The values for RAxML-III are average values over all 10 runs. The best-known maximum likelihood values found for the real alignment data benchmark sets which have all been obtained by RAxML-III or RAxML-SA are marked by bold letters in Table 1. IQPNNI was only executed on the 218_RDPII and 500_ZILLA datasets due to long execution times. The final likelihood value for 218_RDPII was -156348.0 (25138 secs) and -100049.9 (10380 secs) for 500_ZILLA. Those likelihood values of IQPNNI were reached by RAxML-III after an average of 1019 seconds for 218_RDPII and 397 seconds for 500_ZILLA. RAxML-SA required 481 seconds to outperform IQPNNI on the 218-taxon alignment and 446 seconds on the 500-taxon data set.

One important result is that RAxML-SA produced all best-known trees for large alignments containing ≥ 250 sequences. Moreover, RAxML-SA outperforms the average RAxML-III final tree likelihood values on all alignments except 150_ARB. In addition, in all but one case (500_ZILLA) RAxML-III encounters a better tree than the final PHYML tree more rapidly than PHYML. RAxML-SA is naturally slower in some cases, since the likelihood does not improve as fast over time.

It is important to note that the apparently small differences in final likelihood values *are* significant. On the one hand they represent log likelihood values, i.e. differences in order of magnitude. On the other hand the average relative difference of likelihood scores between PHYML trees and the best RAxML-trees is 0.22% over all datasets and 0.36% for the datasets with ≥ 500 sequences. Thus, bearing in mind the remark about a required or desired score-accuracy of 99.99% in the introduction of this paper those differences are significant and the additional computation time to obtain them is not wasted.

5. OpenMP Parallelization

As already mentioned, most of the research carried out in phylogenetics is very algorithmic and theoretical. However, as computation of 5.000-organism ML trees has become feasible, it is also important to consider technical aspects such as cache efficiency and memory consumption. The latter represents an often underestimated problem. The general tendency in phylogenetics is that alignment sizes will grow in both dimensions, i.e. number of organisms *and*

data	PHYML	secs	R-III	secs	R-III(avg)	secs(avg)	RAxML-SA	secs	R-III > PHY	R-SA > PHY
101_SC	-73970.3	70	**-73786.8**	1876	-73797.6	1107	-73791.5	453	29	22
150_SC	-44287.5	86	**-44141.6**	859	-44154.3	996	-44146.7	655	19	9
150_ARB	-76776.9	181	**-76745.7**	1603	-76750.5	1482	-76766.7	1827	101	207
193_V	-64911.1	134	**-64779.1**	2908	-64806.9	2151	-64784.7	907	52	73
200_ARB	-104164.5	250	**-104073.8**	2298	-104074.2	3340	-104076.2	3537	220	399
218_RDPII	-156607.6	219	**-156228.5**	6255	-156268.2	5417	-156260.0	9142	165	235
250_ARB	-130774.4	402	-130690.1	6150	-130720.0	4010	**-130689.6**	10071	309	552
500_ZILLA	-100130.3	246	-99915.9	9247	-99939.1	9362	**-99914.0**	25271	348	472
500_ARB	-251831.9	1108	-251009.3	24089	-251041.0	27416	**-250988.2**	35601	549	499
1000_ARB	-400145.5	5728	-398911.4	94039	-398971.2	118173	**-398904.0**	223085	2707	2895
1663_ARB	-310259.5	5424	-308975.0	210523	-309047.1	181654	**-308952.2**	359799	3479	1234
2025_ARB	-372201.9	7824	-370401.8	242030	-370423.1	243344	**-370342.9**	487745	7354	29294

Table 1. PHYML, RAxML-III, RAxML-SA execution times and likelihood values for real data sets

number of base pairs. For alignments of 1.000 and 10.000 taxa PHYML and MrBayes [11] showed a relatively high memory consumption compared to RAxML-III [24]. A relatively easy way to improve performance of ML programs and partially resolve memory problems at the same time consists in shared memory parallelizations.

The compute-intensive `for`-loops which update the likelihood vectors at each inner node of the tree typically consume up to 90% of overall execution time in ML or Bayesian phylogeny programs. To understand how the individual likelihood vectors are updated consider a subtree rooted at node `p` with immediate descendants `r` and `q` and likelihood vectors `l_p`, `l_q`, and `l_r` respectively. When the likelihood vectors `l_q` and `l_r` have been computed the entries of `l_p` can be calculated—in an extremely simplified manner—as outlined by the pseudo-code below:

```
for(i = 0; i < n; i++)
   l_p[i] = f(g(l_q[i], b_pq),
              g(l_r[i], b_pr));
```

where `f()` is an inexpensive simple function which combines the values of `g(l_q[i], b_pq)` and `g(l_r[i], b_pr)`. The `g()` function however is more complex and computationally intensive since it carries out the evaluation of the nucleotide substitution probabilities. The parameters `b_pq` and `b_pr` represent the branch lengths. Note, that the `for`-loop can easily be parallelized on a fine-grained level since entries `l_p[i]` and `l_p[i + 1]` can be computed independently. An initial parallelization of RAxML-III with OpenMP [17] (called RAxML-OpenMP) showed however that the speedup is extremely hardware-dependent. Table 5 lists the speedup values measured for simulated 100-taxon alignments with lengths of 1.000, 5.000, 10.000, and 20.000 base pairs on a Quad-Opteron and Quad-Itanium2 processor. The bad parallel performance on the Itanium2 processor is most probably due to the memory access architecture of the specific processor which represents a significant bottle-neck. The partially super-linear speedups attained on the Opteron processor are due to the "classic" reason: improved cache efficiency for large data sets. An important advantage of this implementation consists in the minimal effort required for parallelization: the critical `for`-loops in RAxML-III could be parallelized in half a day (the parallelization also applies to RAxML-SA). As already mentioned other programs such as PHYML can easily be parallelized with OpenMP as well. Apart from solving memory problems, the current approach is intended to be used for a hybrid MPI/OpenMP supercomputer implementation. This is due to the fact that RAxML-OpenMP has limited scalability on pure shared memory machines.

number of base pairs	Quad-Opteron	Quad-Itanium2
1.000	2.11	0.88
5.000	3.58	1.38
10.000	4.30	1.51
20.000	4.22	1.42

Table 2. Speedup of RAxML-OpenMP on Opteron and Itanium2 architectures

6. Conclusion, Availability & Future Work

A new algorithm has been introduced which efficiently combines the advantages of simulated annealing and hill-climbing approaches to ML-based phylogenetic inference. A performance study conducted on 40 large simulated alignment datasets shows that RAxML-SA performs reasonably well in comparison to PHYML. In addition, some problems inherent to the usage of simulated data are discussed. Moreover, RAxML-SA is able to out-compete the currently fastest and most accurate programs on real-world data RAxML-III, PHYML, and IQPNNI on large data sets containing ≥ 250 sequences in terms of final likelihood values. In particular, *all* best-known trees for those large alignments have been established with RAxML-SA. Furthermore, the program yields—with one exception—better final trees than the *average* RAxML-III run on real data. Due to the combination of hill climbing and simulated annealing techniques in RAxML-SA the execution times remain comparable to those of strict hill-climbing approaches. A major advantage of RAxML-SA consists in the ability to automatically build consensus trees out of the 100 currently best—topologically distinct—trees of the annealing process on the fly. Thus, RAxML-SA facilitates the computation of biologically reliable and publishable results by *one* single program execution. Finally, Section 5 proposes some efficient technical solutions, in terms of programming time and performance improvement, to further accelerate current state-of-the-art ML phylogeny programs. The complete program package which also includes the standard hill-climbing algorithm is freely available for download as RAxML-V at www-bode.in.tum.de/˜stamatak/.. In order to provide a set of well-studied real alignment data sets as benchmark to the community *all* real alignments used in the current work including the best-known tree-topologies are also available for download. The OpenMP-version of RAxML will become available for download shortly.

Future work will mainly focus on the development of a parallel version of RAxML-SA for hybrid supercomputer architectures.

References

[1] D.A. Bader *et al*. Industrial Applications of High-Performance Computing for Phylogeny Reconstruction. In *Proceedings of SPIE ITCom*, 4528:159–168, 2001.

[2] D. Barker. LVB: Parsimony and simulated annealing in the search for phylogenetic trees. In *Bioinformatics*, 20:274–275, 2004.

[3] V. Cerny. Thermodynamical approach to the travelling salesman problem: An effi cient simulation algorithm. In *J. Optimization Theory Appl.*, 45:41–51, 1985.

[4] M.W. Chase *et al*. Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene rbcL. In *Annals of the Missouri Botanical Garden*, 80:528–580, 1993.

[5] W.E. Day *et al*. The computational Complexity of inferring rooted phylogenies by parsimony. *Math. Bios.*, 81:33–42, 1986.

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

[7] O. Gascuel. BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, 14:685–695, 1997.

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

[9] M. Hasegawa *et al*. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. In *J. Mol. Evol.*, 22:160–174, 1985.

[10] M.T. Holder *et al*. Phylogeny Estimation: Traditional and Bayesian Approaches. *Nature Reviews Genetics*, 4:275–284, 2003.

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

[12] J.P. Huelsenbeck *et al*. Potential Applications and Pitfalls of Bayesian Inference of Phylogeny. *Syst. Biol.*, 51(5):673–688, 2002.

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

[14] S. Kirkpatrick *et al*. Optimization by simulated annealing. In *Science*, 220:671–680, 1983.

[15] W. Ludwig *et al*. ARB: A Software Environment for Sequence Data. In *Nucl. Acids Res.*, 32(4):1363–1371, 2004.

[16] G. Olsen *et al*. fastdnaml: A Tool for Construction of Phylogenetic Trees of DNA Sequences using Maximum Likelihood. *Comput. Appl. Biosci.*, 10:41–48, 1994.

[17] OpenMP: WWW.OPENMP.ORG/DRUPAL.

[18] PAUP project site PAUP.CSIT.FSU.EDU.

[19] PHYLIP downlaod site and list of phylogeny software. EVOLUTION.GENETICS.WASHINGTON.EDU.

[20] A. Rambaut *et al*. Seq-Gen: An application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. In *Comp. Appl. Biosc.*, 13:235–238, 1997.

[21] D. Robinson, L. Foulds. Comparison of weighted labeled trees. In *Lecture Notes in Mathematics*, 748:119–126, Springer, Berlin,1979.

[22] L. Salter, D. Pearl. A Stochastic Search Strategy for Estimation of Maximum Likelihood Phylogenetic Trees. In *Syst. Biol.*, 50(1): 7–17, 2001.

[23] M.J. Sanderson. r8s software: GINGER.UCDAVIS.EDU/R8S.

[24] A. Stamatakis *et al*. Parallel Inference of a 10.000-taxon Phylogeny with Maximum Likelihood. In *Proceedings of Euro-Par 2004*, Volume 3149 of LNCS, 997–1004, Springer Verlag, September 2004.

[25] A. Stamatakis *et al*. New Fast and Accurate Heuristics for Inference of Large Phylogenetic Trees. In *Proceedings of IPDPS2004*, Santa Fe, New Mexico, April 2004.

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

[27] L.S. Vinh and A.v. Haeseler IQPNNI: Moving fast through tree space and stopping in time. In *Mol. Biol. Evol.*, 21(8):1565–1571, 2004.

[28] L.S. Vinh *et al*. PhyNav: A novel approach to reconstruct large phylogenies. In *Proceedings of GfKl conference*, 2004.

[29] T.L. Williams, B.M.E. Moret. An Investigation of Phylogenetic Likelihood Methods. In *Proceedings of BIBE'03*, 2003.

[30] T.L. Williams *et al*. The relationship between maximum parsimony scores and phylogenetic tree topologies. *Tech. Report*, TR-CS-2004-04, Department of Computer Science, The University of New Mexico, 2004.

[31] Z. Yang. Among-site rate variation and its impact on phylogenetic analyses. In *Trends Ecol. Evol.*, 11:367–372, 1996.