Available online at www.sciencedirect.com

SCIENCE DIRECT®

Future Generation Computer Systems 21 (2005) 725–730

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/fgcs

DRAxML@home: a distributed program for computation of large phylogenetic trees

A. Stamatakis[a,*], M. Lindermeier[a], M. Ott[a], T. Ludwig[b], H. Meier[a]

[a] *Department of Computer Science, Technical University of Munich, Boltzmannstr. 3, D-85748 Garching b. München, Germany*
[b] *Department of Computer Science, Ruprecht-Karls-University, Im Neuenheimer Feld 348, D-69120 Heidelberg, Germany*

Available online 5 November 2004

Abstract

Inference of large phylogenetic trees using statistical methods is computationally extremely expensive. Thus, progress is primarily achieved via algorithmic innovation rather than by brute-force allocation of available computational ressources. We describe simple heuristics which yield accurate trees for synthetic (simulated) as well as real data and significantly improve execution time. The heuristics are implemented in a sequential program (RAxML) and a novel non-deterministic distributed algorithm (DRAxML@home). We implemented an MPI-based and a http-based distributed prototype of this algorithm and used DRAxML@home to infer trees comprising 1000 and 2025 organisms on LINUX PC clusters.
© 2004 Elsevier B.V. All rights reserved.

Keywords: High performance bioinformatics; Distributed computing; Phylogenetic tree inference; Maximum-likelihood method

1. Introduction

Within the ParBaum project at the Technical University of Munich, we work on phylogenetic tree inference based on the maximum likelihood method by Felsenstein [1]. A phylogenetic inference starts with a collection of taxa (organsims, sequences) in an appropriately prepared form (multiple alignment) and builds an unrooted binary tree with the sequences at its leaves under an optimality criterion. For elaborate and accurate statistical topology scoring functions such as maximum likelihood the problem is NP-complete. Furthermore, the computation of the topology score itself is computationally expensive.

In this paper, we describe simple heuristics which accelerate the tree optimization process, yield accurate results, and ouperform the currently-to the best of our knowledge-fastest and most accurate programs on real data: MrBayes [3] and PHYML [2]. We first presenteded those new heuristics and related results in [11]. Thus, in this paper we focus on the description and first experimental results of the non-deterministic http-based and MPI-based distributed prototypes of Distributed Randomized Axelerated Maximum Likelihood (DRAxML@home).

1.1. Related work

An excellent comparison of popular phylogeny programs using statistical approaches such as fastD-

* Corresponding author.
E-mail address: stamatak@in.tum.de (A. Stamatakis).

0167-739X/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2004.05.013

NAml [5], MrBayes, PAUP [6], and treepuzzle [13] based on synthetic (simulated) data may be found in [14]. An in-depth analysis of related work can be found in [11].

Distributed algorithms for phylogenetic tree inference have-to our best knowledge-so far only been implemented for parsimony searches by Snell et al. [10] using the CONDOR platform.

2. Heuristics

2.1. Sequential algorithm

“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] [5]). On the other hand, they can start with an initial tree already containing all organisms built by a simpler method such as Neighbor Joining or by random (implemented in [2] [3]). 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 [7] for two reasons.

Firstly, parsimony is related to maximum likelihood under simple evolutionary models [15], such that we can expect to obtain a starting tree with a relatively good likelihood value compared to random or NJ starting trees.

Secondly, 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 Fig. 1 (not all possible moves are shown). The likelihood of the such generated topologies is evaluated and the best tree is kept. If one alternative toplogy 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 Fig. 1 with bold lines RAxML only optimizes the three branches adjactent to the insertion point before computing its likelihood value. Since the likelihood of

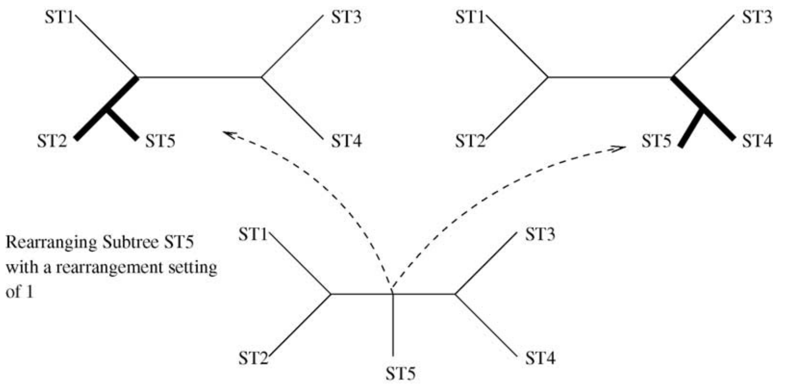

Fig. 1. Rearrangments traversing one node for subtree ST5, branches which are optimized are indicated by bold lines.

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.

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 thistree is kept immediatly and all subsequent subtree rearrangements are performed on the improved topology. This enables rapid optimization of the topology during the initial optimization phase of the algorithm (see below).

2.2. Distributed algorithm

Our Motivation to build a distributed seti@home-like [9] 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 consisits 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, e.g. a compression by factor 31 for a 1000 taxa alignment using gzip. 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 ct, to each worker. Since the subsequent application of topological improvements during 1 rearrangement step is closely coupled we slightly modify the algorithm according to the following observation: Our experiments have shown that subsequent improved topologies occur only during the first rearrangement steps (initial optimization phase). Thereafter, only one alternative topology per rearrangement step improves the likelihood. This behaviour is illustrated in Fig. 2 where we plot the number of improved topologies per rearrangement step for a phylogenetic reconstruction of a 150 taxa tree 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 (final optimization phase). This phase requires the largest amount of computation time, especially with big alignments (>500 organisms, ≈70% of execution time). Thus, during the initial optimization phase we send only one single subtree ID i, $i = 2, ..., \#\text{species} \times 2 - 1$ along with the currently best tree ct to each worker for rearrangements. The worker returns the best tree wt_i obtained by rearranging subtree i to the master. If wt_i has a better likelihood than ct at the master, ct is set to wt_i and distributed to each worker along with the subsequent work (subtree ID) requests.

In the final optimization phase, we reduce communication costs by generating only $5 \times \#\text{workers}$ jobs (subtree ID spans) and entirely avoiding the transfer of tree topologies.

Finally, irrespective of the current optimization phase the best 20 topologies computed by each worker during one rearrengement step are stored in a local worker tree list. When all subtree rearrangements i of one rearrangement step have been completed, each worker conducts branch length optimizations on its best 20 local trees and returns the best topology to the master.

When all workers have branch-length optimized their topologies the master initiates the next rearrangement step until no better tree is found. Due to the required changes to the algorithm the distributed program is non-deterministic, since final output depends

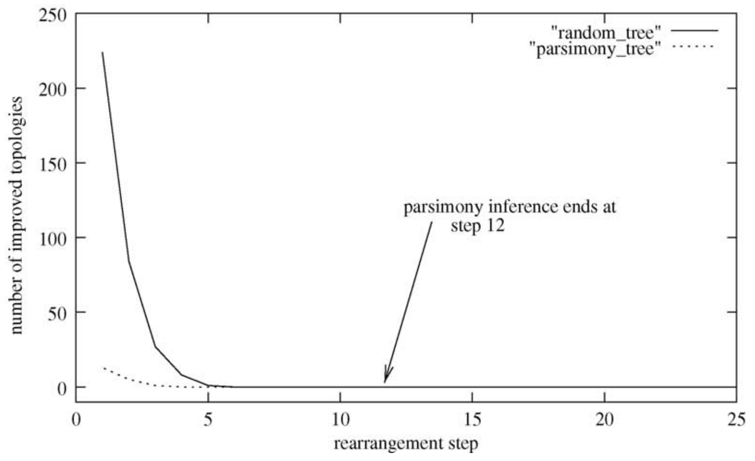

Fig. 2. Number of improved topologies per rearrangement step for a phylogenetic inference with 150 organisms using random and parsimony starting trees.

on the number of workers and on the arrival sequence of results for runs with equal numbers of workers. This is due to the changed implementation of the subsequent application of topological improvements during the initial rearrangement steps which causes a traversal of search space on different paths.

2.3. Technical issues

We will shortly outline some technical issues of our http-based implementation concerning communication, redundancy and security. The communication infrastructure is provided by a http-library which can easily be integrated into the MPI-based prototype by replacing the MPI communication routines. The most expensive part in terms of communication costs is the distribution of the alignment file which is compressed uzing gzip. To obtain redundancy we distribute every subtree rearrangement job twice and use a queue and timeouts to ensure that every job is computed. Furthermore, we have developed a failure protocol which is able to handle temporary master and worker failures. Finally, we deal with the scenario that some workers deliberately return phony trees. If the tree is not in the correct format, this can easily be detected by the routine which reads the tree string. The only problem arises when a worker returns a tree that is in the correct fromat and has a “fake” likelihood (i.e. a likelihood value which is significantly better than the actual likelihood of the topology conatined in the message) which is better than the currently best tree at the master. In this case, the likelihood of that topology is verified by the master process.

3. Results

In order to test our distributed prototypes we executed our MPI-based program for a fixed starting tree on 4, 8, and 16 processors of the LINUX cluster at the RRZE [8] with an alignment containing 1000 organisms (1000_ARB). Furthermore, we conducted an initial test of the http-based prototype on a small LINUX PC cluster at our institute equipped with Intel Xeon 2.4 GHz processors which are interconnected by Infiniband with an alignment of 2025 (2025_ARB) sequences. Due to the design of the distributed implementation we can not expect near-optimal speedups and exactly equivalent results for runs with distinct numbers of processors. The sequential execution on the same processor type for 1000_ARB required 53002 seconds and yielded a final likelihood value of -400970.31. The times at which the distributed program passed the likelihood of the sequential one, the final likelihood values and speedups for 1000_ARB as well as the final result for 2025_ARB are listed in Table 1. For calculating, the speedup we only consider the number of worker processes.

Table 1
Results obtained by the fisrt test with the distributed prototypes of DRAxML@home

Data set	# Processors	Likelihood	Secs	Speedup	Program
1000_ARB	1 (1)	−400970.31	53002	1	Sequential
1000_ARB	4 (3)	−400945.43	17871	2.97	MPI
1000_ARB	8 (7)	−400950.58	10693	4.96	MPI
1000_ARB	16 (15)	−400947.24	7542	7.03	MPI
2025_ARB	10 (9)	−371366.74	141388	Void	http

4. Conclusion & future work

We presented a distributed non-deterministic algorithm for phylogenetic tree inference which is based on heuristics which outperform the currently fastest and most accurate programs for phylogenetic tree inference on real-world data (see [11]) under simple models of site substitution.

Furthermore, we have presented first experimental results for DRAxML@home which demonstrate that computing high-quality phylogenetic trees containing up to 2000 organisms is now feasible on a couple of workstations instead of supercomputers.

Future work will mainly cover the execution of large parallel and distributed production runs and an effort to install and execute DRAxML@home on a greater number of workstations. The overall goal is to compute a first “small” tree of life containing about 10.000 representative organims from the three domains: Archaea, Bacteria and Eukarya.

Acknowledgement

This work is sponsored under the project ID ‘ParBaum’, within the framework of the “Competence Network for Technical, Scientific High Performance Computing in Bavaria” (KONWIHR)

References

[1] J. Felsenstein, Evolutionary trees from DNA sequences: a maximum likelihood approach, J. Mol. E 17 (1981) 368.
[2] S. Guindon, O. Gascuel, A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood, Syst. Biol. 52 (2003) 696.
[3] J.P. Huelsenbeck, F. Ronquist, MRBAYES: Bayesian inference of phylogenetic trees, Bioinformatics 17 (2001) 754.
[4] W. Ludwig, et al., ARB: a software environment for sequence data, Nucl. Acids Res. 32 (2004) 1363.
[5] G. Olsen, et al., fastdnaml: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood, Comput. Appl. Biosci. 10 (1994) 41.
[6] PAUP: paup.csit.fsu.edu, visited May 2003.
[7] PHYLIP: evolution.genetics.washington.edu, visited November 2003.
[8] RRZE: www.rrze.uni-erlangen.de, visited October 2003.
[9] Seti@home: setiathome.ssl.berkeley.edu, visited July 2003.
[10] Q. Snell, et al., Parallel phylogenetic inference, Proceedings of Supercomputing Conference 2000, Dallas, Texas, USA, 2000.
[11] A. Stamatakis, et al., New fast and accurate heuristics for inference of large phylogenetic trees, Proceedings of IPDPS2004, Santa Fe, New Mexico, USA, 2004. Sequential open source code and paper available online at: www.bode.cs.tum.edu/stamatak.
[12] C. Stewart, et al., Parallel implementation and performance of fastdnaml – a program for maximum likelihood phylogenetic inference, Proceedings of Supercomputing Conference 2001, Denver, Colorado, USA, 2001.
[13] K. Strimmer, A.v. Haeseler, Quartet puzzling: a maximum-likelihood Method for reconstructing tree topologies, Mol. Biol. E 13 (1996) 964.
[14] T.L. Williams, B.M.E. Moret, An investigation of phylogenetic likelihood methods, Proceedings of BIBE’03, Bethesda, Maryland, USA (2003) 79–86.
[15] C. Tuffley, M. Steel, Links between maximum likelihood and maximum parsimony under a simple model of site substitution, Bull. Math. Biol. 59 (1997) 581.

Alexandros Stamatakis received his diploma in Computer Science from the Technical University of Munich in 2001. His studies included internships abroad at the Ecole Normale Superieure de Lyon, National Technical University of Athens, Eurocontrol Experimental Center (Paris), and Instituto de Salud Carlos III (Madrid). Since October 2001 he works as PhD Student at the Lehrstuhl fuer Rechnertechnik und Rechnerorganisation of the technische

Universitaet Muenchen. His main research focus lies on parallel and distributed systems and algorithms for computation of large phylogenetic trees.

Markus Lindermeier studied computer science at the Technical University of Munich from 1992–1998. In June 1998, he joined the Lehrstuhl fuer Rechnertechnik und Rechnerorganisation as PhD student. He worked on load managment in distributed systems and received his doctoral degree in 2002. He currently works for the BMW group in the competence center for IT-Architectures.

Michael Ott is a student of Computer Science at the Technische Universitaet Muenchen since October 1999. He currently works on his diploma thesis and will receive his diploma in autumn 2004. Since summer 2001, he works as a student assistant at the Lehrstuhl fuer Rechnertechnik und Rechnerorganisation at the Technische Universitaet Muenchen. His main research interests are parallel and distributed applications and architectures.

Thomas Ludwig received his habilitation degree from Technische Universität München in Munich, Germany, where he worked for 13 years in the field of parallel computing with a focus on load balancing, development tools, and cluster and tool infrastructures. He also conducted research in the field of parallel programming, namely with computer tomography and bioinformatics. Since 2001, he is a professor for computer science at the Ruprecht-Karls-Universität Heidelberg in Heidelberg, Germany. His current research focus is in the field of high performance parallel input/output systems for cluster environments.

Harald Meier was borne on the 19th of May 1962 in Augsburg (Bavaria, Germany) He studied Biology at the Technische Universitaet Muenchen, and received the diploma degree in 1993. At the Department for Microbiology he performed his PhD in the area of molecular identification of bacteria and phylogenetic treeing, and received his Dr. rer. nat. in 1997. From 1997 until 1999, he held a postdoc position at the GSF-National Research Center in Neuherberg, on development of molecular methods for insuring the microbial safety of bottled mineral waters. In 1999/2001, he performed postgraduate studies in Computer Sciences in Munich and Bioinformatics at the University Heidelberg. Since 2001, he is head of the group ’Applied High Performance Bioinformatics’ at the LRR at the Institute for Informatics, Technische Universitaet Muenchen. His main research topics are the molecular sequence analysis, molecular phylogeny, design and analysis of DNA-probes and DNA-Microarrays, and high performance computing.