Workflows with Model Selection: a Multilocus Approach to Phylogenetic Analysis

J. Álvarez-Jarreta[1], R. Blanco[1], E. Mayordomo[1]

[1] Departamento de Informática e Ingeniería de Sistemas (DIIS).
Group of Discrete Event Systems Engineering (GISED).
Aragon Institute of Engineering Research (I3A).
University of Zaragoza, María de Luna 1, 50018, Zaragoza, Spain.
Tel. +34-976762472, Fax +34-976761861
e-mail: jorgeal at unizar.es, robertob at unizar.es, elvira at unizar.es

Abstract

The workflow model of description and execution of complex tasks can be of great use to design and parallelize scientific experiments, though it remains a scarcely studied area in its application to phylogenetic analysis. In order to remedy this situation, we study and identify sources of parallel tasks in the main reconstruction stages as well as in other indispensable problems on which it depends: model selection and sequence alignment. Finally, we present a general-purpose implementation for use in cluster environments and examine the performance of our method through application to very large sets of whole mitochondrial genomes, by which problems of biological interest can be solved with new-found efficiency and accuracy.

Keywords: computational phylogenetics, model selection, workflows, mitochondrial DNA, multilocus data.

1 Introduction

Phylogenetics is a prominent branch of the discipline of bioinformatics, founded on the fundamental driving force of life: evolution. Its goal is to ascertain the relations between living organisms, extant and extinct, and determine the history and course of the diversity of life at large. Despite the shortcomings of imperfect information a great scientific corpus of knowledge has been amassed over the past decades: advances in software techniques and computer architecture are continually expanding the frontiers of what is practicable as opposed to what, due to excessive computational requirements, is not. Inference of phylogenies belongs to the ample class of interesting problems that resist efficient treatment for their very combinatorial nature. Moreover, the use of models of evolution that reflect the specific patterns of change observed in each dataset is crucial for obtaining realistic phylogenies but has been avoided for large datasets due to its associated computational cost.

Parallel execution is one in the arsenal of techniques that can be used to reduce total running times of (in particular) costly algorithms, putting to good use the copious computational resources that are available in the form of independent processors interconnected by gargantuan communication networks. Efforts in this direction have been focused on fine-grain parallelization of standard algorithms and application of algorithmic engineering techniques to improve implementations ((Olsen et al., 1994; Stamatakis, Ludwig and Meier, 2005) are representative examples). Some of these undertakings have harvested remarkably good performance measures, but nonetheless typical algorithms were not designed with an eye on concurrency, and the number of independent tasks at a given moment is limited, as is their individual load ─which may hinder simple assignment schemes or constrain its use to tightly coupled, low-latency networks. Master-worker schemes dominate these approaches.

On the other hand, workflows are an abstract formalism to describe complex tasks composed of related subtasks. This systematization developed naturally in manufacturing and business environments. When applied to scientific experimentation, not only are they found to structure and document experiments very fittingly, but given suitable specifications and implementations they form the base for automated experiment execution environments (Georgakopoulos, Hornick and Sheth, 1995). Many software projects have been developed to this end, including several specialized in bioinformatics applications (Oinn et al., 2004). Unfortunately, they are not well suited to expressing and managing arbitrary levels of parallelism, though with some effort they can be coaxed into describing it to a degree. Yet their highly interactive nature becomes their undoing for these purposes. Previous work in workflows for phylogenetics has followed this philosophy (Bowers et al., 2008).

While low-level and interactive execution environments have distinct advantages, both can benefit from lower computational costs, the former being used by our proposed workflows, which at the same time can be integrated into the latter. We advocate an effective use of: a) known sources of independent, potentially concurrent tasks; and b) known or inferred biological information which simplifies the general case of uninformed algorithms, and offers solutions of higher quality in less time.

With this in mind, this paper has two major objectives: firstly, to unveil existing high-level concurrency in traditional approaches to the phylogenetic reconstruction problem, including partitioning methods that increase granularity and improve running times while allowing further biological insight (e.g., different genes and loci in multilocus studies evolve differently); secondly, to design and implement arbitrarily scalable, fully automated workflows to put these ideas efficiently into practice, with an emphasis on modularity, ease of maintenance and problem integration (most notably, model selection).

2 Material and Methods

2.1 Problem Decomposition

We focus our study on the canonical reconstruction problem in computational phylogenetics. Given a set of sequences S, suitably aligned, our goal is to produce a tree T which satisfies (or approximates) a certain optimality criterion. The connection between S and T is given by a labeling of the leaves of the tree by the sequences of the set. The fundamental dimensions of the problem, which govern algorithmic complexity, are the number of sequences s, shared by S and T, and the length of these sequences l, hidden in the tree. By l we usually designate the total number of cladistic characters of the dataset, and equivalently the length of the alignment.

Obviously, at some point it is necessary to solve these kinds of simple problems, for which there exists a plethora of algorithmic methods, themselves poorly scalable in general. We need not concern ourselves with these at this point, though each may allow certain low-level improvements, which are completely compatible with the high-level workflows that are discussed here. Only one aspect must be contemplated now: the method of statistical evaluation (Holder and Lewis, 2003). This customary addition is imposed by the desirable assessment of such traits as robustness and quality of the results. In most methods, a number of statistical replicates *r* is provided to a sampler (possibly alongside additional parameters), which generates an equal number of derivative alignments. Each of these constitutes an independent problem of the same magnitude as the original, solved independently and condensed with the rest in the final solution. In this we find a first level of data-independent concurrency.

Additionally, we can identify in model selection (Sullivan and Joyce, 2005) a precedent task that is apt for deterministic treatment. Whereas the parameters for use with the selected tree algorithm may be furnished by the user, it is generally convenient to employ selection procedures that evaluate a wide range of models *M* and elect that which best appears to fit the aligned data. Once again, we find a simple workflow composed of as many tasks as models under consideration *m*; each of these is independent from the rest and, when all models have been assessed, their results are harvested and the best pick is decided upon. This process takes place after alignment and before any of the resampled instances may commence execution.

So much for problem-independent concurrency. Nonetheless, an oft-disregarded source of independent tasks can be found in the data themselves and, most importantly, is exploitable by automated means. To boot, biological data are far from unstructured and, as a matter of fact, complex multilocus studies are becoming increasingly common. Therefore, it will be beneficial to make use of what information is previously known about data. In this light, preclassification in accordance with established facts or hypotheses ─notwithstanding the trial of these by whatever means necessary─ becomes of great use to produce partitioned datasets with twofold benefits: the generation of independent tasks and the reduced size of these. We will now examine the effects of this proposal in both fundamental dimensions.

Firstly, let us consider the nature of cladistic characters, represented by *l*. Despite the homogeneous nature of sequence alignments, genomes are actually composed of coding (genes) and non-coding regions, often subject to evolutionary pressure in different form and intensity. Consequently, an alignment should be divided in subsets of columns corresponding to each self-contained genetic unit, say *g* in number. For this, it suffices for the alignment to contain an annotated sequence with unit thresholds, and use these to perform the splitting. Each subalignment thus generated can be processed as described above (sampling and model selection), and finally combined with its companions by some suitable model of coalescence (Degnan and Rosenberg, 2009).

Secondly, whereas sequences represent individual organisms (*s* in terms of problem complexity) and it is the task of phylogenetic analysis to ascertain their relationships, it may well be possible to easily identify groups of evolutionarily related sequences in advance (haplogroups in our case), and treat these as indivisible units for the purposes of tree construction. To this end, a hierarchical classification scheme can be used to classify each sequence as belonging to one of *h* groups of subtrees, which can be independently built and integrated by resort to supertree algorithms (Bininda-Emonds, Gittleman and Steel, 2002). We have previously shown how this technique offers great improvements in large, representative datasets (Blanco et al., 2010). Obviously, both strategies can be combined for greater effect.

It must be noted that the multiple sequence alignment problem, which precedes all others, can profit by the application of the same partitioning principles that have just been expounded. In fact, both sources of partition information —an annotated sequence for *l* and a sequence classifier for *s*— are ordinarily applied to individual sequences, aligned with the reference sequence pairwise.

2.2 Workflow Design

The number and variety of independent tasks thus uncovered is certainly great and offers many possibilities for concurrent execution. The nature of these divisions is both simple and homogeneous, consisting of generation steps, multiple-instance execution steps, and combination steps, where the middle stage comprises the bulk of the computational load. The different types of tasks are further arranged in a nested fashion: reduction operations generate groups of related, though independent, datasets, until the basic problems are solved and the classifying cascade is reversed by the appropriate combination of summarizing algorithms (see Figure 1).

From these considerations we propose a modular workflow based on the definition of reusable black boxes for each significant layer of concurrent work. The construction we present is typical, though by no means unique, and variants are simple to produce as needed. Each layer is easily overridden by supplying trivial classifiers if needed.

The basic problem of computation of phylogenies consists of a core stage and three parallel layers, as follows.

Tree algorithms. The basic computation stage may in fact be workflow-like in form, due to combination of several programs (distance methods are typical examples) or to low-level parallel implementations. Whatever the case, this “box” can be assumed to take an alignment *A* and a set of parameters and produce a tree *T*.

Statistical sampling. The fundamental parallel layer comprises the statistical sampling and concurrent solution of a number of basic problems, followed by the application of a consensus algorithm. Its interface is similar to that of the basic box, except that it must be provided the number of replicates *r* which determine the total magnitude of its associated workload.

Gene trees. Preprocessing on *l* operates on simple sequence alignments and generates a number of subalignments to be subsequently sampled according to the split instructions represented by *S*, which is required additionally to the parameters of its subordinate tasks. Note that a set of models *M* rather than a single model μ may be supplied to choose the most adequate parameters for each gene, as will be explained shortly. As with the other nested boxes, its purpose is the production of a single tree *T* that explains the alignment *A*.

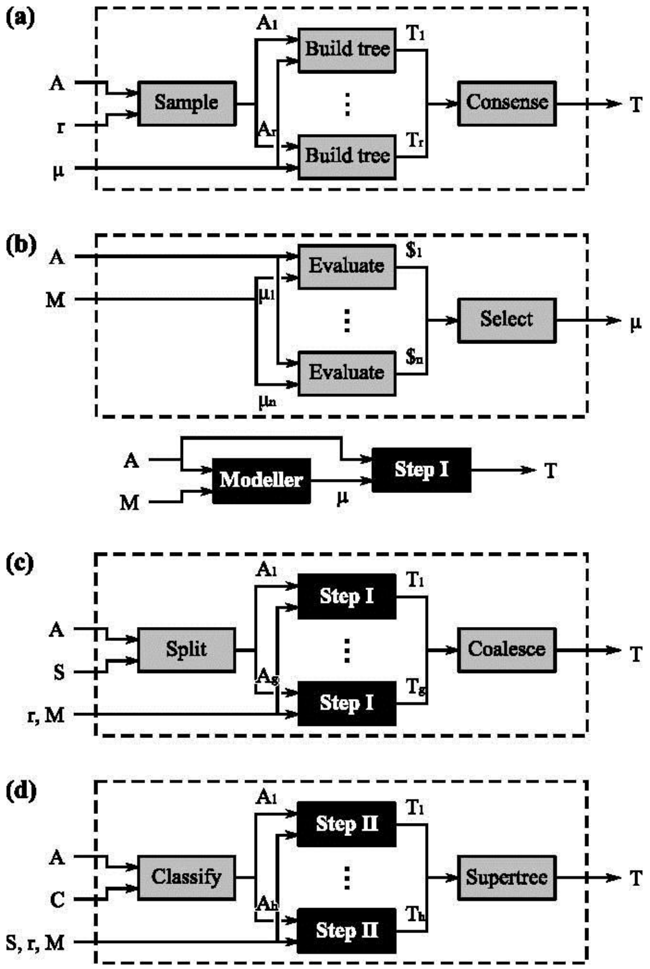

Figure 1: Bottom-up hierarchy of concurrent levels with their nestings and interactions with related problems: **(a)** concurrent level 1 (statistical sampling); **(b)** concurrent model selection and integration with Level 1; **(c)** concurrent level 2 (gene trees); and **(d)** concurrent level 3 (supertrees).

Supertrees. Treatment of s is achieved by a hierarchical classifier C charged with generating the inputs for each subproblem, which are then passed on to the gene tree stage. It should be noted that C may be added as an input to the supertree gatherer. The rest of its parameters are passed on to the next nested layer, as usual.

Furthermore, the following precondition problems are an obvious target for workflow integration, as well.

Model selection. The purpose of model selection, unlike that of previous layers, is the selection of a model μ from a set of candidates M, according to a given alignment A, for which some measure of the fitness of each model must be provided by the scoring program; after that, a selector evaluates these results (possibly balanced against the complexity of each model) and emits its choice. This computation is customarily included as part of Level 1, or rather immediately before it.

Sequence alignment. Levels 2-3 can be adapted to the sequence alignment problem with minimal changes. They operate on unaligned sets of sequences instead of the alignments that they are tasked with generating, each classifier dividing these collections of sequences in either shorter or smaller ones. Workflow structure remains the same, and only executable programs (or "boxes") need to be replaced.

2.3 Implementation Issues

Each task to be performed on a part of the input (partial alignment, model analysis, gene and subtree reconstruction) can be computed independently from the rest.

The resulting parallelism is ideally suited to the use of a cluster as a generally available and highly flexible execution environment.

Out of the available alternatives we have selected Condor: a management system specialized in high-throughput computing. One of the tools that make Condor an optimal choice is DAGMan (Directed Acyclic Graph Manager) (Couvares et al, 2006): a meta-scheduler that allows to design an order relationship between processes. The design of our system has no cycles, so there is a direct translation between it and a directed acyclic graph (DAG). Moreover, DAGMan offers the possibility of designing a DAG's where one node can be a new DAG, so the nodes after this one will have to wait until the whole DAG ends correctly to start their work. This technique, called *DAGMan within DAGMan*, contributes greatly to the black-box properties of our system.

Condor workflows are generated automatically from the inputs and their classifiers. For a detailed discussion of technical considerations, most importantly regarding process size and job scheduling, see (Álvarez, 2010).

3 Results

Let us start with a simple estimation of complexity. As described above, our system divides the input alignment into h haplogroups in the first step, and then, each haplogroup is divided into g genes. In the following step, m models are analyzed for each gene, selecting the best one and finally, r bootstraps are executed for the selected model. Consequently, the number of jobs involved in our system totals $h\cdot(g\cdot(m+r+3)+2)+2$.

We have tested the system with an alignment of 4895 complete sequences of real human mitochondrial DNA (mtDNA) produced by the ZARAMIT project for comprehensive phylogenetic studies (Blanco and Mayordomo, 2009). Here we have $h=26$ (the number of non-empty haplogroups in a basic classification), $g=38$ (the 37 genes in human mtDNA, plus the control region), and $m=88$ (the set of models included in the current version 0.1.1 of the application *jModelTest* (Posada, 2008) and most frequently used in the systematics studies). Replacing these values in the equation above we obtain a total of $988\cdot r+89962$ jobs.

Feasibility tests ($r=15$) and full-scale tests ($r=200$, well in the range $[100, 1000]$ of typical bootstrap figures for studies in systematics) have been performed. Table 1 summarizes real and sequential (estimated) time costs of system runs and other relevant data. Note that for estimating the sequential cost of the system we have assumed that each job needs 4 minutes to schedule its execution on average.

No. of bootstraps	No. of jobs	CPUs available	CPUs used (mean)	Sequential cost (days)	Cluster cost (days)	Speedup
15	104782	400	200	291	3	97
200	287562	600	250	799	12	66.6

Table 1: Results of system execution in the cluster with different number of bootstraps.

Now, we shall estimate the running time when maximum parallelism is achieved, that is, when all jobs are executed simultaneously in separate cluster nodes. The largest partition will provide the worst-case scenario; in our case, this corresponds to gene MT-ND5 (1812 aligned base pairs) and haplogroup M (582 sequences). We have computed all 88 models in a scientific workstation with Intel Core 2 Duo processor and 8 GB of RAM, to determine which model is the costliest in terms of time in the aforementioned gene and haplogroup. TVM+I+G was the worst, with a cost of 1 hour. Therefore, the critical path of our system should cost about 2 hours and 20 minutes (including intermediate jobs and scheduling time). This means that in a sufficiently large cluster, our system will take just that amount of time to complete the whole phylogenetic study.

To complete the evaluation of partial speedups commenced in (Blanco et al., 2010), we have compared model selection in our system with the cost of *jModelTest*, the most commonly used (sequential) software. With just a subset of 200 sequences of our main alignment, *jModelTest* took more than 17 hours to run all 88 models, while our system took just above 1 hour. More detailed results will be presented shortly.

4 Discussion

We have developed a system where a divide and conquer methodology has been applied to design workflows (employing the black-box principle of transparency) that integrate model selection and phylogenetic reconstruction, and so can deal with extensive phylogenetic analysis in an efficient way. The system has been tested with very large datasets of mtDNA and accepts any type of biological data as inputs. In addition, the criteria for input partitioning can be customized in order to reflect the nature of inputs; of course, the number of bootstraps can be modified as well. The system yields speedups higher than 50 compared to its sequential equivalent in large phylogenetic studies; we have obtained great improvements in the model selection phase alone compared with common specific-purpose tools like *jModelTest*.

Finally, for future developments we will aim for improvements of the speedup achieved, which appears to degrade with problem size. We will also seek ways to integrate input retrieval and alignment process as preliminary steps in our system. Further improvements on the computational cost are expected due to the inner parallelism of this kind of processes.

5 Acknowledgments

This research was supported by the I3A Fellowship Program and the Spanish Ministry of Science and Innovation (MICINN) [TIN2008-06582-C03-02]; and the Spanish Ministry of Education [AP2008-03447].

We want to thank the Aragon Institute of Engineering Research (I3A) for their support with cluster Hermes and in particular Antonio Sanz for his assistance.

References

Álvarez, J. (2010). *Análisis teórico-práctico de métodos de inferencia filogenética basados en selección de modelos y métodos de superárboles.* Master's thesis.

Bininda-Emonds, O.R.P., Gittleman, J.L. and Steel, M.A. (2002). The (super)tree of life: procedures, problems and prospects. *Annual Review of Ecology, Evolution, and Systematics*, 33:265-289.

Blanco, R. and Mayordomo, E. (2009). ZARAMIT: A system for the evolutionary study of human mitochondrial DNA. In: *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. Springer, pp. 1139-1142.

Blanco, R. et al. (2010). Scalable phylogenetics through input preprocessing. In: *Advances in Bioinformatics*. Springer, pp. 123-130.

Bowers, S. et al. (2008). Kepler/pPOD: scientific workflow and provenance support for assembling the tree of life. In: *Provenance and Annotation of Data and Processes*. Springer, pp. 70-77.

Couvares, P. et al. (2006). Workflow management in Condor. In: *Workflows for e-Science*. Springer, pp. 357-375.

Degnan, J.H. and Rosenberg, N.A. (2009). Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in Ecology & Evolution*, 24:332-340.

Georgakopoulos, D., Hornick, M. and Sheth, A. (1995). An overview of workflow management: from process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3:119-153.

Holder, M.T. and Lewis, P.O. (2003). Phylogeny estimation: traditional and Bayesian approaches. *Nature Reviews Genetics*, 4:275-284.

Oinn, T. et al. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20:3045-3054.

Olsen, G.J. et al. (1994). fastDNAml: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *Computer Applications in the Biosciences*, 10:41-48.

Posada, D. (2008). jModelTest: phylogenetic model averaging. *Molecular Biology and Evolution*, 25:1253-1256.

Stamatakis, A., Ludwig, T. and Meier, H. (2005). RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21:456-463.

Sullivan, J. and Joyce, P. (2005). Model selection in phylogenetics. *Annual Review of Ecology, Evolution, and Systematics*, 36:445-466.