

*A multi-objective evolutionary approach to  
phylogenetic inference problems:  
Initial experiences porting PhyloMOEA to  
ParadisEO framework*

Waldo Cancino — L. Jourdan — E-G. Talbi

N° 0366

June 2009

Thème NUM



*R*apport  
*de recherche*



## A multi-objective evolutionary approach to phylogenetic inference problems: Initial experiences porting PhyloMOEA to ParadisEO framework

Waldo Cancino , L. Jourdan , E-G. Talbi

Thème NUM — Systèmes numériques  
Équipes-Projets DOLPHIN

Rapport de recherche n° 0366 — June 2009 — 30 pages

**Abstract:** Several phylogenetic reconstruction methods have been proposed in order to find the best tree that represents the evolutionary history of species analyzed. Most of these methods define an optimality criterion for the evaluation of possible solutions. However, different criteria may lead to dissimilar phylogenies, which often conflict with each other. In this context, a multi-objective approach can be useful since it could produce a set of optimal trees according to multiple criteria. PhyloMOEA proposes a multi objective approach to phylogenetic inference using maximum parsimony and maximum likelihood criteria. However, in many aspects, PhyloMOEA is a very basic tool with limited expandability. The availability of metaheuristic and bioinformatics reusable frameworks provides an excellent opportunity to increase PhyloMOEA capabilities. In this report we describe the development of a new parallel PhyloMOEA version using the ParadisEO framework

**Key-words:** phylogenetic inference, multi-objective optimization, ParadisEO

# Une approche multi-objective pour le probleme de inference phylogenetique

## Les experiences initiales de développement de PhyloMOEA utilisant la plateforme ParadisEO

**Résumé :** Plusieurs méthodes de reconstruction phylogénétique ont été proposées dans le but de trouver le meilleur arbre décrivant au mieux l'histoire de l'évolution des espèces. La plupart de ces méthodes définissent un critère d'optimalité pour l'évaluation de toutes les solutions réalisables. Cependant, différents critères peuvent mener à des phylogénies dissemblables qui sont souvent en conflit entre elles. Dans ce contexte, une approche multi-objectif peut être utile puisqu'elle peut produire un ensemble d'arbres optimaux selon plusieurs critères. Cependant, sur plusieurs aspects, PhyloMOEA est un outil très simple avec peu d'extensibilité. La disponibilité de plateformes réutilisables de métaheuristiques et de bioinformatique fournit une excellente opportunité d'améliorer les capacités de PhyloMOEA. Dans ce rapport, nous décrivons le développement d'une nouvelle version parallèle de PhyloMOEA utilisant la plateforme ParadisEO.

**Mots-clés :** inference phylogénétique, optimisation multi-objectif, ParadisEO

## 1 Introduction

Phylogenetic inference is one of the central problems in computational biology. It consists in finding the best tree that explains the evolutionary history of species from a given dataset. Various phylogenetic reconstruction methods have been proposed in the literature. Most of them use one optimality criterion (or objective function) to evaluate possible solutions in order to determine the best tree.

On the other hand, several researches [27, 33, 56] have shown important differences in the results obtained by applying distinct reconstruction methods to the same input data. In this context, a multi-objective approach can be a relevant contribution since it can search for phylogenies using more than one criterion and produce trees which are consistent with all employed criteria. Handl et al. [26] discussed the current and future applications of multi-objective optimization in bioinformatics and computational biology problems.

One of the first studies that models the phylogenetic inference as a multi-objective optimization problem (MOOP) was proposed by the author of this report [5]. In this approach, the multi-objective approach used the maximum parsimony [20] and maximum likelihood [15] as optimality criteria. In order to solve this problem, a multi-objective evolutionary algorithm (MOEA), called *PhyloMOEA* was developed.

The *PhyloMOEA* output is a set of distinct solutions representing a trade-off between the considered criteria. Results show that the found trees are statistically consistent with trees found by maximum parsimony and maximum likelihood analyses performed separately. Moreover, the clade supports obtained from the trees found by *PhyloMOEA* approximate, in general, the clade posterior probabilities of trees inferred by Bayesian inference methods [5].

One of the main drawbacks of first version *PhyloMOEA* is the difficult to incorporate new features to the proposed approach. It is due mainly the program was developed from scratch without using mature metaheuristic and/or bioinformatic frameworks available. *ParadisEO* [3] is a open-source object-oriented framework for metaheuristic design. Its clear separation between problem specific and independent parts of the solutions method provides maximum design and code reuse. Thus, porting *PhyloMOEA* code to use *ParadisEO* is a suitable alternative to extend the original algorithm with minimal coding effort. The main objective of this report is to describe thoroughly the development of *PhyloMOEA* version using *ParadisEO*. During this process, several advantages and difficulties were identified and solved.

This report is organized as follows. Section 2 provides relevant background information about phylogenetic inference and two phylogenetic reconstruction methods: maximum parsimony [21] and maximum likelihood [15]. Section 3 presents a brief summary of the MOO applications in phylogenetic inference. Section 4 describes the development of *PhyloMOEA* using the *ParadisEO* framework. Section 5 is focused on the parallelization of *PhyloMOEA* using the distributed/parallel modules in *ParadisEO*. Section 6 describes the the experiments involving the new *PhyloMOEA* version developed. Finally, Section 7 presents conclusions and proposes future work.

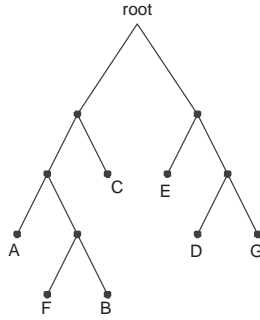


Figure 1: A rooted tree.

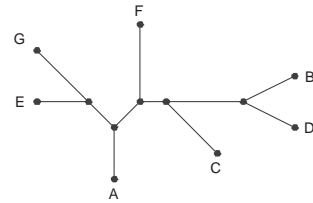


Figure 2: An unrooted tree.

## 2 Phylogenetic Reconstruction

Phylogenetic analysis studies the evolutionary relationships among species. The data used in this analysis usually come from sequence data (nucleotide or aminoacid sequences), morphological features, or other types of data [17]. Frequently, researchers only use data from contemporary species due the information about past species is unknown. Consequently, the phylogenetic reconstruction is only an estimation process since it is based on incomplete information [54].

The evolutionary history of species under analysis is often represented as a leaf-labelled tree, called phylogenetic tree. The actual species (or taxa) are represented by the external nodes of the tree. The past species (ancestors) are referred by internal nodes of the tree. Nodes are connected by branches which may have an associated length value, representing the evolutionary distance between the nodes connected by the branch. It is important to stress that a phylogenetic tree is a hypothesis (of many possible ones) concerning the evolutionary events in the history of species.

A phylogenetic tree can be rooted or unrooted. In a rooted tree, there is a special node called root, which defines the direction of the evolution, determining ancestral relationships among nodes. An unrooted tree only shows the relative positions of nodes without an evolutionary direction. Figures 1 and 2 show a rooted and an unrooted tree, respectively.

The main objective of the phylogenetic inference is the determination of the best tree that explains the evolutionary events of the species under analysis. Several phylogenetic reconstruction methods have been proposed in the literature. Swofford et al. [54] separated phylogenetic reconstruction methods into two categories:

1. Algorithmic methods, which use well-defined steps to generate a tree. An important feature of these methods is that they go directly to the final solution without examining many alternatives in the search space. Consequently, the solutions are quickly produced by these methods. Clustering approaches like NJ [43] and UPGMA [36] are in this category.
2. Optimality criterion methods, which basically have two components: an objective function (optimality criterion) and a search mechanism. The

objective function is used to score each possible solution. The search mechanism walks through the tree search space in order to find the best scored tree according to the used criterion. Optimality methods are slower than algorithmic methods, however, they often provide more accurate answers [27]. Examples of optimality criterion methods are maximum parsimony [20], maximum likelihood [15] and least squares [6].

One of the main problems in phylogenetic inference is the size of the tree search space which increases exponentially in function of the number of taxons. In the case of optimality criterion methods, this means that the search mechanism requires heuristic techniques, which are able to find adequate solutions in reasonable running time for large or even moderate datasets. Exhaustive and exact search techniques can also be employed, although their use is constrained to problems with a small number of species.

Sections 2.1 and 2.2 present a brief review of the criteria employed in this study: maximum parsimony and maximum likelihood.

## 2.1 Maximum Parsimony

The parsimony principle states that the simplest hypothesis concerning an observed phenomenon must always be preferred. Parsimony methods search for a tree that minimizes the number of character state changes (or evolutionary steps). This tree, called maximum parsimony tree, refers to the simplest explanation of the evolutionary history for the species in a given dataset [17].

Let  $D$  be a dataset containing  $n$  species. Each specie has  $N$  sites, where  $d_{ij}$  is the character state of specie  $i$  at site  $j$ . Given tree  $T$  with node set  $V(T)$  and branch set  $E(T)$ , the parsimony score of  $T$  is defined as [54]:

$$PS(T) = \sum_{j=1}^N \sum_{(v,u) \in E(T)} w_j \cdot C(v_j, u_j), \quad (1)$$

where  $w_j$  refers to the weight of site  $j$ ,  $v_j$  and  $u_j$  are, respectively, the character states of nodes  $v$  and  $u$  at site  $j$  for each branch  $(u, v)$  in  $T$  and  $C$  is the cost matrix, such that  $C(v_j, u_j)$  is the cost of changing from state  $v_j$  to state  $u_j$ . The leaves of  $T$  are labelled by character states of species from  $D$ , i.e., a leaf representing  $k$ -th species has a character state  $d_{kj}$  for position  $j$ . The following properties can be noted from Equation (1):

1. Parsimony criterion assumes independence of sites, i.e., each site is evaluated separately;
2. The calculation of the parsimony score only takes into account the tree topology. Thus, the parsimony criterion does not incorporate other information, like branch lengths.

There are several variants of the parsimony criterion. One of the simplest is the Fitch parsimony [20], which assumes a unitary cost matrix such that  $C_{xy} = 1$  if  $x \neq y$ ; otherwise  $C_{xy} = 0$ . The Fitch and even other more complex variants of parsimony can be even generalized for arbitrary cost matrix and restrictions of state changes [44].

Given a tree  $T$ , it is necessary to determine the character states of its internal nodes such that  $PS(T)$  is minimized. This is also known as the small parsimony problem. In the case of the Fitch parsimony, a post-order traversal in  $T$  is enough to minimize  $PS(T)$  (this procedure is known as Fitch algorithm [20]). In the case of generalized parsimony, the small parsimony problem can be solved by applying the Sankoff algorithm [44].

Having defined an algorithm to minimize  $PS(T)$  for a given tree  $T$ , we should determine the tree  $T^*$  such that  $PS(T^*)$  is the minimum for all tree search space. The problem of finding  $T^*$  is called large parsimony problem, which was proved to be NP-hard [17]. However, several heuristic techniques have been proposed to overcome such a difficulty [23].

## 2.2 Maximum Likelihood

Likelihood is a widely-used statistical measurement. It evaluates the probability of a hypothesis giving rise to the observed data [54]. Thus, a hypothesis with higher probability is preferred to one with lower probability. The likelihood of a phylogenetic tree, denoted by  $L = P(D|T, M)$ , is the conditional probability of the sequence data  $D$  given a tree  $T$  and an evolutionary model  $M$ , which contains several parameters related to tree branch lengths and a sequence substitution model [17]. Two assumptions are necessary to compute likelihoods:

1. Evolution at different sites is independent;
2. Evolution from different tree lineages is independent, i.e., each subtree evolves separately.

Given a tree  $T$ ,  $L(T)$  is calculated from the product of partial likelihoods from all sites:

$$L(T) = \prod_{j=1}^N L_j(T), \quad (2)$$

where  $L_j(T) = P(D_j|T, M)$  is the likelihood at site  $j$ . The site likelihoods can also be expressed as:

$$L_j(T) = \sum_{r_j} C_j(r_j, r) \cdot \pi_{r_j}, \quad (3)$$

where  $r$  is the root node of  $T$ ,  $r_j$  refers to any possible state of  $r$  at site  $j$ ,  $\pi_{r_j}$  is the frequency of state  $r_j$ , and  $C_j(r_j, r)$  is the conditional likelihood of the subtree rooted by  $r$ . More specifically,  $C_j(r_j, r)$  is the probability that everything that is observed from node  $r$  to the leaves of  $T$ , at site  $j$ , given  $r$  has state  $r_j$ . Let  $u$  and  $v$  be the immediate descendants of  $r$ , then  $C_j(r_j, r)$  can be formulated as:

$$C_j(r_j, r) = \left[ \sum_{u_j} C_j(u_j, u) \cdot P(r_j, u_j, t_{ru}) \right] \times \left[ \sum_{v_j} C_j(v_j, v) \cdot P(r_j, v_j, t_{rv}) \right], \quad (4)$$

where  $u_j$  and  $v_j$  refer to any possible state of nodes  $u$  and  $v$ , respectively.  $t_{rv}$  and  $t_{ru}$  are the lengths of the branch connecting node  $r$  to nodes  $v$  and  $u$ ,

respectively.  $P(r_j, u_j, t_{ru})$  is the probability of changing from state  $r_j$  to state  $u_j$  during evolutionary time  $t_{ru}$ . Similarly,  $P(r_j, v_j, t_{rv})$  is the probability of changing from state  $r_j$  to state  $v_j$  at time  $t_{rv}$ . Both probabilities are provided by the evolutionary model  $M$ .

An efficient method to calculate  $L$  was proposed by Felsenstein [15] using a dynamic programming approach, where  $L$  is obtained by a post-order traversal in  $T$ . Usually, it is convenient to work with logarithmic values of  $L$ , then Equation (2) results in:

$$\ln L(T) = \sum_{j=1}^n \ln L_j(T). \quad (5)$$

The likelihood calculation presented in this section assumes that sites evolve at equal rates. However, this assumption is often violated in real sequence data [61]. Several among site-rate variation (ASRV) approaches can be incorporated in model  $M$ . One of the most employed ASRV approaches is the discrete-gamma model [60] where variables rates at sites follow a  $\Gamma$  distribution discretized in  $N_{cat}$  categories. When the discrete-gamma model is considered, Equation 3 becomes:

$$L_j(T) = \sum_{k=1}^{N_{cat}} \sum_{r_j} \rho_k \pi_{r_j} C_j(r_j, r, \omega_k), \quad (6)$$

where  $\omega_k$  and  $\rho_k$  are the rate and the probability of the  $k$ -th category, respectively. Usually, all categories are assumed to have the same probability ( $\rho_k = 1/N_{cat}$ ). Values  $\omega_k$  are obtained from a discretized  $\Gamma$  distribution. In order to calculate the conditional likelihood for the  $k$ -th category, denoted as  $C_j(r_j, r, \omega_k)$ , Equation 4 is modified as follows:

$$C_j(r_j, r, \omega_k) = \left[ \sum_{u_j} C_j(u_j, u, \omega_k) \cdot P(r_j, u_j, t_{ru} \omega_k) \right] \times \left[ \sum_{v_j} C_j(v_j, v, \omega_k) \cdot P(r_j, v_j, t_{rv} \omega_k) \right]. \quad (7)$$

Several studies [27, 56, 59] have pointed out that the use of ASRV models can improve the results of the likelihood inference. However, ASRV models also increase the computational cost of the likelihood calculations.

In order to maximize  $L$  for a given tree  $T$ , it is necessary to optimize the parameters of model  $M$  (i.e: branch lengths and parameters of the substitution model chosen), which can be achieved using classical optimization methods [17]. Finding the maximum likelihood tree in the search space is a more difficult problem. Moreover, only heuristic approaches [25, 34, 35, 46] are feasible for large or even moderate datasets.

Another problem regarding likelihood calculation is the numeric precision for large datasets involving more than 100 species [24, 58]. Consider Equation (4), if large datasets are involved, the conditional likelihoods  $C_j(u_j, u)$  and  $C_j(v_j, v)$  produce very small values. Thus, the resulting  $C_j(r_j, r)$  value may

not be representable on a computer generating an underflow error. In this case, a scaled conditional likelihood, denoted as  $C_j^*(r_j, r)$ , is calculated according to the following expression:

$$C_j^*(r_j, r) = \left[ \sum_{u_j} \frac{C_j(u_j, u)}{\rho_j(u)} \cdot P(r_j, u_j, t_{ru}) \right] \times \left[ \sum_{v_j} \frac{C_j(v_j, v)}{\rho_j(v)} \cdot P(r_j, v_j, t_{rv}) \right], \quad (8)$$

where  $\rho_j(u) = \max_{u_j} \{C_j(u_j, u)\}$  and  $\rho_j(v) = \max_{v_j} \{C_j(v_j, v)\}$  are the scaling factors of the conditional likelihoods for nodes  $u$  and  $v$  at site  $j$ , respectively. Using  $C_j^*(r_j, r)$ , the scaled likelihood, denoted by  $L_j^*(T)$ , is obtained as shown in the following Equation:

$$L_j^*(T) = \frac{1}{\rho_j(u)\rho_j(w)} \times \sum_{r_j} \pi_{r_j} C_j(r_j, r). \quad (9)$$

The summation expression in Equation (9) is exactly  $L_j(T)$  from Equation (3). Applying natural logarithm to both sides of Equation (9) and replacing terms, we obtain:

$$\ln(L_j(T)) = \ln(L_j^*(T)) + \ln(\rho_j(u)) + \ln(\rho_j(v)). \quad (10)$$

This procedure enables the calculation of the original likelihood value by the addition of the logarithms of scaling factors.

### 3 Multi-objective approaches to phylogenetic inference

The use of various phylogenetic reconstruction methods can produce different results for the same input data. Huelsenbeck [27] tested the main phylogenetic approaches for simulated datasets containing four species. In this study, most methods performed successfully, however, under some conditions, methods failed to find the true tree producing different answers. Thus, the selection of the reconstruction method is a crucial step in phylogenetic analysis.

Rokas et al. [41] pointed out other sources of incongruity in phylogenetic analysis: the datasets used and evolutionary assumptions concerning data. Frequently, data for a group of species can come from different sources. The use of conflicting datasets in phylogenetic reconstruction, combined or separated, may produce different trees

Poladian and Jermin [40] propose an MOO approach to deal with conflicting data. The authors built two artificial datasets of four DNA sequences which were analyzed using the maximum likelihood criterion. The authors explored the nature of conflicting trees in the Pareto-front obtained. Solutions along Pareto-front reveal how topologies and likelihood values change. These results are useful to summarize all solutions in representative topologies without loss of information about Pareto front or fitness landscape. The authors also note the importance of the four-species problem for phylogenetic reconstruction methods based on quartets of species [53].

Coelho and Von Zuben [10] propose an multi-objective Artificial Immune System (AIS) approach, called omni-aiNet, to the reconstruction of phylogenetic trees. The omni-aiNet is employed to find a set of Pareto-optimal trees that represent a trade-off between the minimum evolution and the least-squares criteria. Both criteria employs distance data to evaluate tree topologies.

Cancino and Delbem [5] propose a MOO approach for phylogenetic reconstruction using maximum parsimony [20] and maximum likelihood [15] criteria. This approach, called PhyloMOEA, is based on the NSGA-II model [13]. Further details of PhyloMOEA and its porting to use ParadisEO framework are given in the next section.

## 4 Porting PhyloMOEA to ParadisEO

PhyloMOEA consists of two basic components: the evolutionary algorithm and the phylogenetic solver. The EA component was initially based on the MOEA library developed by Xiaming Xen, obtained from the EMOO repository [11]. Several new features and bug corrections were performed to this library during PhyloMOEA development. The most important features added were the incorporation of MOEA comparison metrics and the implementation of new algorithms like NSGA-II and SPEA2 [63]. As results of these efforts, a new library called LibMOEA was born<sup>1</sup>. A Parallel MOEA cooperative component was lately added, although it was not extensively tested.

The phylogenetic solver component is responsible for the representation of phylogenetic trees, sequence data management, likelihood and parsimony evaluations and branch length optimization. It is important to point out that this component was developed from scratch due to the lack of documentation regarding implementation details in the main phylogenetic programs available. Besides, the phylogenetic solver can be executed independently from the MOEA component. This component separation allows that other metaheuristics libraries can be used with the phylogenetic solver.

ParadisEO<sup>2</sup> is an open-source meta-heuristic framework which assists the development of program solvers. It provides several reusable components which aims to accelerate and minimize the coding efforts. ParadisEO has four main modules:

- ParadisEO-EO: which implements population-based metaheuristics.
- ParadisEO-MO: focused in single solution-based metaheuristics.
- ParadisEO-MOEO: aimed for multi-objective metaheuristic and their hybridization.
- ParadisEO-PEO: for parallel and distributed metaheuristics (also covers their hybridization).

ParadisEO-EO and ParadisEO-MOEO were the main modules involved in PhyloMOEA port. During this process, new components, performance optimizations and bug fixed were contributed to ParadisEO. The following section describes porting of the main PhyloMOEA components.

<sup>1</sup>LibMOEA and PhyloMOEA first versions can be found at <http://sourceforge.net/projects/libmoea/>

<sup>2</sup><http://paradisEO.gforge.inria.fr/>

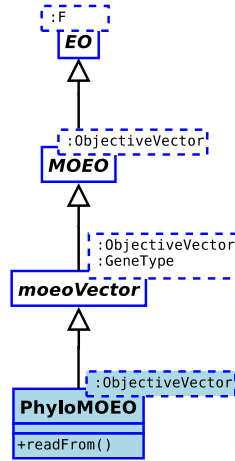


Figure 3: UML diagram for PhyloMOEA solution representation.

#### 4.1 Internal Encoding

Phylogenetic trees are usually represented using an unrooted tree data structure. An internal node is represented as a circular linked list, where each node has a pointer to its adjacent nodes [1, 17]. The degree of an internal node defines the number of elements in the list.

On the other hand, PhyloMOEA employs a standard graph structure provided by the Graph Template Library (GTL) [22]. GTL facilitates the implementation of genetic operators and the storage of additional information, such as branch lengths. Furthermore, parsimony and likelihood criteria can operate on rooted or unrooted trees.

The solution representation was implemented in the `PhyloMOEO` class. This class is derived from the `MOEO` template class provided by `ParadisEO-MOEO`. It is also necessary to define a copy constructor and an attribution operator specifically designed to manipulate phylogenetic tree objects. Besides, this class contains a pointer to the phylogenetic tree class already implemented in previous PhyloMOEA versions. Figure 3 shows the UML diagram for `PhyloMOEO` class.

#### 4.2 Initial Solutions

The `ParadisEO-EO eoInit` class handles the initialization of the first population. The `PhyloMOEA` initial trees are generated from either random or user-defined trees. The generation of random trees is implemented in the `PhyloMOEORanInit` class while the user-defined tree initialization is handled by the `PhyloMOEONewickinit` class. Both classes are derived from `eoInit`.

The initial user-defined trees can be provided before `PhyloMOEA` execution. These trees must follow the Newick tree format [18], commonly used in phylogenetic inference tree programs. Initial user-defined trees are read by the `eoPop` class, which handles the population of solutions. For each tree read, `eoPop` calls the `readFrom` method, which parses an input tree in Newick format. Thus, it was necessary this method in the `PhyloMOEO` class. Figure 4 shows the inheritance for `PhyloMOEA` tree initialization classes.

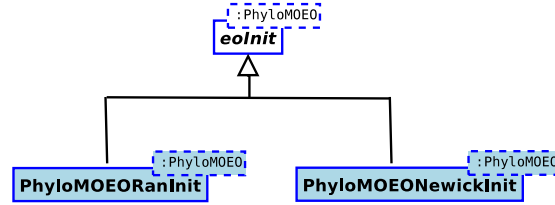


Figure 4: UML diagram for PhyloMOEA solution initialization.

PhyloMOEA can generate initial random trees; however, these trees are poor estimations of the maximum parsimony and likelihood trees. In this case, the PhyloMOEA's convergence is severely affected. In order to overcome this drawback, the initial solutions are provided by maximum likelihood, maximum parsimony and bootstrap analysis [16], which are performed before PhyloMOEA's execution. This strategy is usually employed by other GA-based phylogenetic programs [31, 34].

### 4.3 Evaluation

PhyloMOEA calculates parsimony and likelihood scores of the unrooted trees using the Fitch and Felsenstein algorithms [20]. The class `PhyloMOEOWalk`, responsible for function evaluations, was derived from the `moeoWalkFunc` template class provided by ParadisEO-MOEO.

The evaluation of likelihood scores performs a significantly number of floating point operations. In large datasets, such evaluation is time consuming. Several alternatives exist in order to distribute these calculations among several processors [47]. The parallelization of parsimony and likelihood calculations is detailed in Section 5.

### 4.4 Variation operators

The recombination operator implemented in PhyloMOEA is the same operator proposed in [35]. It combines a subtree from two parent trees and creates two new offspring trees. Given trees  $T_1$  and  $T_2$ , this operator performs the following steps:

1. Prune a subtree  $s$  from  $T_1$ ;
2. Remove all leaves from  $T_2$  that are also in  $s$ ;
3. The offspring subtree  $T'_1$  is obtained by regrafting  $s$  to an edge randomly chosen from  $T_2$ .

The second offspring, denoted as  $T'_2$  is created in a similar way: prune a subtree from  $T_2$  and regraft it in  $T_1$ . Figure 5 illustrates this operator.

There are three well-known topological modifications used in phylogenetic inference [54]: NNI (nearest neighbor interchange), SPR (subtree pruning and regrafting) and TBR (tree bisection and reconnection). NNI was employed in PhyloMOEA, since it performs fewer topological modifications than the others. This mutation operator performs the following steps:

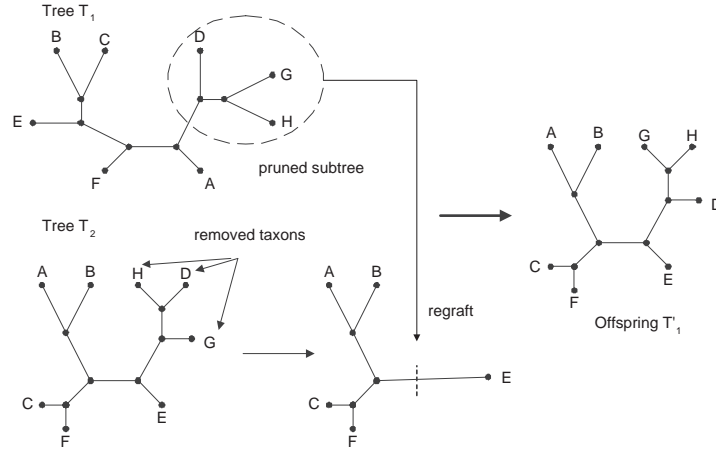


Figure 5: Example of the crossover operator.

1. Choose an interior branch whose connected nodes  $i, j$  define two pairs of neighbors:  $A, B$  adjacent to  $i$  ( $A, B \neq j$ ) and  $C, D$  adjacent to  $j$  ( $C, D \neq i$ );
2. Execute a swap of two nodes taken from each pair of neighbors.

Figure 6 illustrates the NNI mutation operator. This operator also modifies branch lengths in order to improve the tree likelihood value. Some branches, chosen at random, have their lengths multiplied by a factor obtained from a  $\Gamma$ -distribution [35].

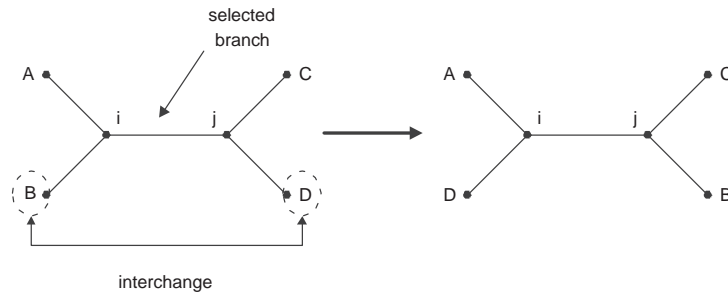


Figure 6: Example of NNI mutation operator.

The recombination operator class, called `PhyloMOE0Cross`, was derived from `eoQuadOp` while the `PhyloMOE0Mutate` mutation operator class was derived from `eoMonOp` class. Both base classes are provide by `ParadisEO-EO` module.

#### 4.5 MOEA models

The first version of `PhyloMOEA` is based on the NSGA-II algorithm implemented in `LibMOEA`. However, `ParadisEO-MOEO` provides other MOEA models including `SPEA2` and `IBEA` [62]. The new `PhyloMOEA` implementation can

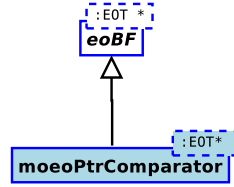


Figure 7: UML diagram for the moeoPtrComparator class.

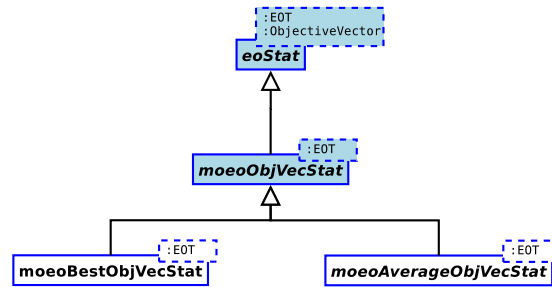


Figure 8: UML diagram for ParadisEO-MOEO statistic classes.

choose between NSGAII and IBEA as MOEA models. The comparison of both alternative models are detailed in Appendix A

The crowding diversity calculation in NSGA-II algorithm depends of several sorting operations applied in the population. These sort calls performs swap operation (attribution operators), which involves copy of individuals. In simple solutions representations, the effects of these swaps are not noticeable. However, PhylMOEA class has a costly attribution method. Thus, the several sorting operations greatly increases the execution time of the diversity calculation.

In order to overcome this problem, the sorting operations are only performed in a list of pointers to the members of population individuals. The `moeoPtrComparator` class allows to compare the MOEO objects addressed by pointers. Figure 7 shows the UML diagram for this class.

#### 4.6 Statistics and output files

PhylMOEA stores statistics concerning the population during iterations. ParadisEO-EO provides the `eoStat` base-class, which is derided by several other classes that calculates several population stats. However, the ParadisEO-MOEO module does not provide any classes that deals with populations of MOEO objects. The average and best scores of each objective function are required by PhylMOEA to store population statistics every given generations. The `moeoObjVecStat` base-class, a specialization of `eoStat`, was created to compute statistics for each objective function separately. Finally, `moeoBestObjVecStat` and `moeoAverageObjVecStat`, derived from `moeoObjVecStat` store that best and average scores for each objective function. The UML diagram for the new aforementioned classes is showed in Figure 8. Note that the aforementioned classes are general purpose, and can be employed in for other problems.

PhylMOEA creates several output files that stores Pareto-optimal solutions, population snapshots, algorithm evolution statistics, and clade support

rates for trees found. The PhyloMOEA output needs are covered mainly by the `ParadisEO-EO` class `oFileMonitor`. An additional class, `eoSingleFileMonitor` was added in order to save the population statistics every number of iterations in a single file.

The PyloMOEA `ParadisEO-EO` port was successfully finished and all features of the previous PhyloMOEA version were maintained. The next step involved in the PhyloMOEA version was to exploit the parallel capabilities of the `ParadisEO-PEO` module. In the following section, the parallel PhyloMOEA implementation is described.

## 5 Parallel strategies for PhyloMOEA using `ParadisEO-PEO`

### 5.1 Phylogeny and parallelism

The increasing availability of large sequence data proposes new challenges for high performance environments. In this context, parallel and distributed computing can be used not only to speedup the search, but also to improve the solution quality, search robustness and to solve larger problem instances[55]. Large search space and costly evaluation function (i.e: likelihood) are the main bottlenecks of many serial phylogenetic inference programs. In this context, there are several parallel approaches for found in the literature. The purpose of this section is to summarize these studies empathizing the most important contributions in terms of algorithm design and parallelism. Most of the heuristic tree search methods tree follow hill-climbing, divide-and-conquer or stochastic strategies. We adopt this classification and present the main approaches in the literature according to the type of search method implemented. It is important to point out that although exact parallel search methods, they can not be applied to large problem instances.

A typical hill-climbing tree search begins with a small tree taxon tree. The other taxons are added sequentially selecting the best insertion location. Each time a taxon is added, several round of local tree rearrangements are performed until no improvement is reached. For example a SPR rearrangement step consist of prune all possible subtrees and regraft them in all possible positions. Exhaustive branch length optimization is performed for each topology generated. If a new best tree is found during a specific SPR move, this new tree is used as starting point for a new round of SPR moves. The best tree obtained from rearrangements is used for the next taxon addition. Once the last taxon is added, a final round of tree SPR modifications are performed against the complete tree. `Dnaml` (part of the `PHYLIP` inference package [19]) and `fastDNaml` [52] (based on `DNaml`) are typical examples of phylogenetic software that follows this strategy.

th the add taxon and the topological modifications generates several topologies that must be evaluated, which is a very time consuming process. The parallel version of `fastDNaml` [52] addressed this problem using a master/slave scheme where topological evaluations are distributed across worker processes. The results reports near linear speedup up to 64 processors. However, the authors points out the synchronization necessary after each add taxon and topological modification steps as a limiting speedup factor. Ceron et al. [7] propose

a parallel implementation of the DNAmI program, which uses a master/slave approach similar to the fastDNAmI. The authors studies how to break the sequential dependency in topological rearrangements: it is necessary to evaluate each rearrangement to check if it produces a better tree after continue with the other rearrangements. If a tree with higher likelihood score is found, it becomes the new start tree for future search. A carefully running time analysis shows that in most of the cases, the local rearrangements rarely find trees with better likelihood scores. Thus, it is safe to perform local rearrangements independently. The performance penalty incurred in the cases where a better tree is found is small according to the authors. Experiments were performed using several benchmark datasets with varying numbers of taxa and sites. The results shows that the parallel DNAmI scales well in a wide range of multiprocessor environments although the speedup obtained was lower than one obtained in parallel fastDNAmI [39].

One possible modification to this strategy is to begin with a initial tree obtained by a fast maximum parsimony or clustering method. After, this initial tree is modify iteratively by topological rearrangements until no improvement is reached. RAXML (initially based fastDNAmI) incorporate this strategy and includes two important differences in the tree rearrangement step [49]:

- After a SPR movement, RAXML only optimizes the tree branch adjacent to the regrafting position. This method, called lazy subtree rearrangements, allows to quickly evaluates promising topologies that can improve the maximum likelihood tree found so far. After a complete rearrangement step, a complete branch length optimization is performed on the best 20 tree obtained.
- If during a SPR movement an best tree is found, it is keep and used as new starting point only for the subsequent tree 6movements. This means that, only a fraction of SPR movements are evaluated from this new tree instead of a new round of all possible rearrangements.

The parallel version of RAXML follows a master/slave scheme [48]. The master is responsible for sending the initial topology and distribute LSR tasks to the workers. In serial RAXML version there is a sequential dependency between the a LSR movement and the subsequent rearrangements when a best tree is found. This dependency was broken in the parallel version at a cost of modify the algorithm behavior. Let  $w_0$  and  $w_1$  be two workers that perform LSR movements for subtrees  $i$  and  $i + 1$ . If the worker  $w_0$  detects a better topology, the new best tree will be transferred to  $w_1$  with some delay. In the meantime, worker  $w_1$  performs LSR modifications on the old best tree. This introduces some non-deterministic search in the parallel approach producing different results than the serial version [1]. However, this effect is important only in the firsts iterations of the algorithm where the best topology is constantly improved. At the end of the LSR tasks, the workers sends a list of the best topologies found to the master. The master receives the list for each worker, determine the overall best topologies and sends the best trees to the workers for branch length optimization. Finally, the master collects results from the workers and start a new algorithm iteration until no topology improvement is achieved. Using a medium sized cluster (32-64 Intel Xeon nodes), the parallel RAXML was able

to infer a phylogenetic tree for a 10.000 taxon sequence data being one of the largest tree inferred using maximum likelihood to date.

Recent versions of RAxML [47, 50] also explore a fine-grained parallelism: site independence of likelihood allows that partial likelihoods, log likelihoods and branch length optimization to be calculated separately for each site. In a recent study, Stamatakis and Ott [47] compares the scalability of the OpenMP, Pthreads and MPI approaches for the aforementioned calculations in multi-core, shared memory and massively parallel architectures. The Pthreads/MPI follows a master/slave scheme where the master distributes commands for compute partial likelihoods, optimization and sum log likelihood operations to the workers/threads. The threads/workers perform these operations for a fraction of sites the sequence data. In the case of the MPI implementation, data is partitioned among the different nodes allowing to deal with very big datasets. The master collects results for each worker/threads using reduction operations. On the other hand, the OpenMP approach was implemented using compiler directives. The proposed model were tested using benchmark datasets comprising thousands of sites on several target architectures. The results indicates good scalability and even superlinear speedup (mainly in very large datasets) for the three approaches due to increase cache efficiency. Moreover, customized RAxML versions exists for specific heterogeneous multicore processors like IBM Cell [51], Graphics Units Processors (GPUs) [8]. RAxML [45] is one of the fast, accurate and actively developed maximum likelihood inference programs.

Divide and conquer heuristics decompose the phylogenetic inference problem in small subproblems and ensemble the partial solutions in a single tree. Quartet-based [53] and disk-covering methods [1, 28] are examples of this class of methods. Vinh and von Haeseler [57] propose the IQPNNI algorithm which combines NNI arrangements<sup>6</sup> and quartet methods in order to quickly explores the tree search space. IQPNNI begins generating a initial tree by applying NNI rearrangements over a tree generated using the BIONJ method. Then, some taxa are separated for the tree and reinserting in new positions. These rearrangements are evaluated by the important quartet puzzle (IQP) algorithm, which determines a subset of possible quartets (called important quartets). The best tree obtained by applying IQP are further optimized by NNI movements. The IQP+NNI steps are repeated until a specific stopping criterion is reached. Parallel versions of IQPNNI (piQPNNI) using a pure MPI implementation [37] and an hybrid MPI/OpenMP scheme [38] were also proposed. Both versions use a master/slave approach that distribute the IQP+NNI evaluations across workers. The master collects the best tree and update the best tree in workers in non-blocking way, i.e. best tree in some workers may be outdated. In the hybrid parallel version, OpenMP is used to distribute the site calculation of the likelihood function. The authors test both piQPNNI versions with several benchmark datasets and found that the hybrid version scales better in most of the cases.

Evolutionary computation approaches were also successfully applied to phylogenetic inference problem [34, 35, 64]. Zwickl [64] proposed a GA approach called GARLI (Genetic Algorithm for Rapid Likelihood) which was developed in order to find the maximum likelihood tree for moderate and large sequence data. The initial GARLI initial population containing either randomly generated or user provided trees. An rough optimization is performed on the parameters of the sequence model of evolution and the branch lengths of the initial trees.

During its iterations, GARLI executes the usual selection and mutation genetic operators and preserves the best tree found to the next generation. The mutation operators produce changes in tree topology (NNI, SPR), branch-length and parameters of the model of evolution. Several remarkable improvements in the topological search and branch length optimization are introduced in GARLI in order to reduce computational time required to perform the aforementioned tasks:

- Adaptive tuning of mutation operator rates, which allows to empathize the operators that have been more effective over recent generations.
- Costly mutation operators as optimization of the sequence evolutionary model parameters are conducted each number of iterations.
- Selective branch length optimization process: for each topological modification, GARLI only optimize these branches that are directly affected by the rearrangement. Additional branch length optimizations are iteratively executed in the neighborhood of the affected branches only when these modifications result in likelihood increase greater than a threshold. The threshold used is decreased during the GARLI iterations. Finally, a final optimization round is performed on all branches altered in the previous steps.

GARLI finishes its execution when no improvements are obtained during topological modifications and update intervals or when the threshold used on branch length optimization reaches its minimum value.

The parallel version of GARLI, named P-GARLI, follows a cooperative model where several nodes running a serial version of GARLI collaborates in the search of the best tree. Thus, the aim of the P-GARLI is not focused on speedup the execution time by distributing the calculations performed in the serial algorithm. However, the power of parallel processing is used to improve the results in terms of solution quality. One of the P-GARLI nodes is named master while the other ones are denominated slaves. The role of the slaves is to execute the GARLI serial version and communicates the best tree found to the master at each predetermined time interval. In addition, slaves receives from the master a tree which replaces the worst solution in the local worker population. The master is responsible for the coordination of the search performed by the slaves and also maintain the best trees found by them. The trees received from the slaves does not suffer modifications but can participate in the generation of new individuals of the master population. The recombination operator choses an individual for the master population and one solution received from the slaves. A common clade for both parents is chosen and copied to the second parent. The offspring tree is modified by an full branch length optimization process. The level of master/slave communications can be predefined or dynamically modified by P-GARLI. In the experiments, P-GARLI found test tree than the serial GARLI version on several benchmark datasets. Moreover, GARLI compares favorably against RAXML in datasets comprising until one thousand of species.

In the phylogenetic reconstruction problem, topological search and tree evaluations are the main tasks to be parallelized. In the case of maximum likelihood criteria, Bader et al. [1] classify parallel strategies in three groups:

- Fine-grained: the evaluation of the maximum likelihood function is performed in parallel. Equation 3 shows that likelihood calculations for each site can be done independently. The evaluation of site likelihood can be distributed using threads (for SMP machines [50]), message passing (for cluster environments [52]) or a combination of these two schemes [1, 47].
- Coarse-grained: the topology search and the tree evaluation are distributed. For instance, alternative tree modifications can be performed and evaluated in a master/slave model [47, 52]. This approach not only allows to evaluate the search space more thoroughly but also reduces significantly the search execution time.
- Job-level: multiple searches can be performed using different starting trees. It is also possible to run multiple bootstrap analyses.

It is important to point out that this classification also applies to other parallel approaches using other optimality criteria like maximum parsimony [9] or Bayesian inference [42].

## 5.2 Metaheuristics parallelism

Population-based metaheuristics, as PhyloMOEA, can be parallelized at three levels [55]:

- Algorithmic-level: where independent and collaborative algorithms are running in parallel.
- Iteration-level: in this model, each iteration of metaheuristic is parallelized in order to speedup the algorithm and reduce the search time.
- Solution-level: focused on the parallelization of a single solution.

It is recognized that the costly floating points operations involved in likelihood calculations are the main bottleneck of maximum likelihood inference programs [47]. A profile analysis of PhyloMOEA serial code reveals that around 90% of its execution time is consumed evaluating the likelihood function. Similar results were obtained from other maximum likelihood inference programs like RAxML [46] and PHYML [25].

In order to overcome this problem, PhyloMOEA was parallelized at iteration and solution levels. It is also important to note that, at these levels, the algorithm behavior is not altered. Further details of PhyloMOEA parallelization are given in the following sections.

## 5.3 Parallelizing PhyloMOEA at iteration level

The first step for parallelizing PhyloMOEA was carried at iteration level using a master/slave scheme. The master process is responsible for distributing solutions from the population to be evaluated by the worker processes. The slaves performs the likelihood and parsimony evaluations and return the results to the master. Once all evaluations are collected by the master, it performs the the selection, recombination and generates the new population.

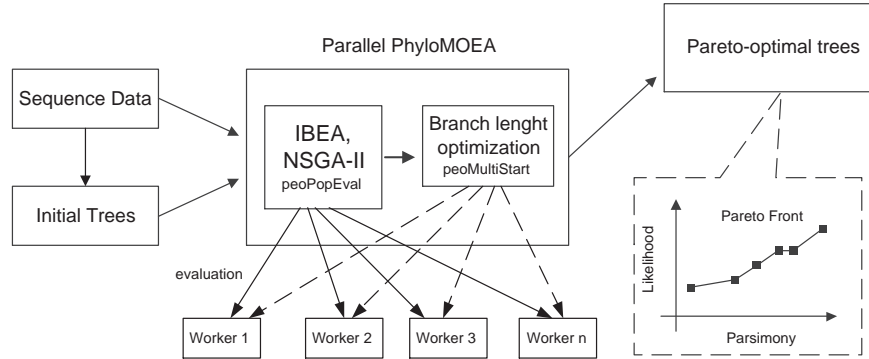


Figure 9: Iteration level parallelization in PhyloMOEA

The parallelization of PhyloMOEA using ParadisEO was quite straightforward. The ParadisEO-PEO component provides the class `peoPopEval` which takes into account all communications needed to distribute the function evaluations across the workers.

The branch length optimization of the solutions, performed at the end of PhyloMOEA execution, can be also distributed. For this purpose the `peoMultiStart` class, provided also for ParadisEO-PEO, was used. Figure 9 illustrates PhyloMOEA-P, the iteration level parallel approach described here.

#### 5.4 Parallelizing PhyloMOEA at solution level

A fine-grained parallelism was implemented in PhyloMOEA by distributing the calculation of the evaluation functions. Both likelihood and parsimony criteria assume site-independence, i.e, the evaluation of each site can be performed separately (see Equations 1 and 5 for parsimony and likelihood criteria).

In PhyloMOEA implementation of parsimony and likelihood functions, the per site evaluations are collected in a loop. As OpenMP (OMP) is well-suited for automatic loop parallelization, it was used to develop the multi-thread version of the parsimony and likelihood functions. Thus, different sites are evaluate in separate threads on multi-core processors. The code changes needed to implement such approach are minimal, as only additional compiler directives are required.

Figure 10 shows PhyloMOEA-OMP, the solution parallel level approach explained in this section.

## 6 Preliminary results

This section describes the performed tests and analysis of the results for both PhyloMOEA-P and PhyloMOEA-OMP.

### 6.1 Solution level scalability

In order to determine the speedup obtaining by PhyloMOEA-OMP version, we compute only the likelihood function on a solution set. We used the following datasets taken for the literature [47]:

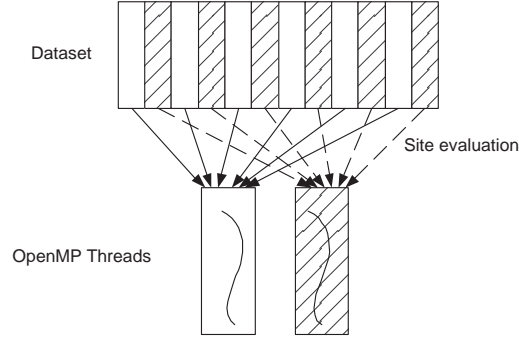


Figure 10: Solution level parallelization in PhyloMOEA

Figure 11: Speedup for AMD Opteron 2168. Figure 12: Speedup for Intel Xeon E530.

- *d50\_5000*, *d50\_50000* and *d50\_500000*: each dataset contains 50 taxa with 5000, 50000 and 500000 sites, respectively.
- *d250\_5000*, *d250\_50000* comprising 250 taxa with 50000 and 500000 sites, respectively.
- *d500\_5000* with 500 taxa and 5000 sites.

PhyloMOEA-OMP version was executed ten times for each dataset. Each execution evaluates 20 randomly generated trees. The target 64-bit quad-core target architectures for each experiment were the AMD Opteron 2168 Q (2.6Ghz) and the Intel Xeon E530 (2.66Ghz). The average execution time for all runs was used to calculate the speedup in each dataset.

Figures 11 and 12 show the speedup for the Opteron and Xeon architectures, respectively. These values in both architectures are around 1.8 for 2 threads and 3.2 for 4 threads in most of the cases.

The scalability of the OMP implementation is restricted by the number of processor cores. In the experiments, we only used up to 4 threads due the resources restrictions. The impact of the number of taxa and the site number in the speedup appears to be not noticeable. However, explore scalability in more cores can provide a better insight about the PhyloMOEA-OMP behavior.

The solution level parallelism implemented here can be hybridized with the iteration level one. For example, instead of run PhyloMOEA-P using 4 workers for solution evaluations, it is possible to use PhyloMOEA-OMP with 1 worker and 4 threads. The behavior of the iteration level and the mixed iteration/solution level parallelism is explored in the next section.

## 6.2 Iteration level scalability

PhyloMOEA serial and parallel versions (P and OMP) was executed using the following datasets:

1. The *rbcL\_55* dataset comprises 55 sequences (each sequence has 1314 sites) of the *rbcL* chloroplast; gene from green plants [35];

2. The *mtDNA\_186* dataset contains 186 human mitochondrial DNA sequences (each sequence has 16608 sites) obtained from The Human Mitochondrial Genome Database (mtDB) [29];
3. The *RDPII\_218* dataset comprises 218 prokaryotic sequences of RNA (each sequence has 4182 sites) taken from the Ribosomal Database Project II [12];
4. Finally, the *ZILLA\_500* dataset includes 500 *rbcL* sequences (each sequence has 1428 sites) from plant plastids [25].

The PhyloMOEA initial population was obtained by parsimony, likelihood and bootstrap analyses described elsewhere [4]. Table 1 shows the parameters of PhyloMOEA used for the experiments. Due to walltime restrictions in Grid5000, PhyloMOEA was running using a small number of iterations and the final branch length optimization was not performed.

Table 1: Parameters used by PhyloMOEA in the experiments.

Parameter	Value
Generations	50
Population size	50
Crossover rate	0.8
Mutation rate	0.05
Mutation operator	NNI
Substitution model	HKY85

The PhyloMOEA-P scalability was measured using 2, 4, 8, 12 and 16 workers (we denote these variants as 2w, 4w, 8w and 16w). The PhyloMOEA-OMP version was tested with 1 worker (2 and 4 threads, denoted by 1w-2t, 1w-4t), 2, 3 and 4 workers (4 threads each, denoted by 2w-8t, 3w-12t and 4w-16t). Figures 13, 14, 15, 16 show the speedup values obtained by both PhyloMOEA versions for *rbcL\_55*, *mtDNA\_186*, *RDPII\_218* and *ZILLA\_500*, respectively.

For all datasets, the 2w, 4w and 12w variants have better speedup values than the 1w-2t, 1w-4t and 3w-8t configurations. However, the 4w-16t is faster than the 4w variant for all datasets except the *ZILLA\_500*. A super-linear speedup is achieved mostly for PhyloMOEA-P version for 2w and 4w while PhyloMOEA-OMP version only reach similar results with the 2w variant. The speedup increasing rate of the PhyloMOEA-P is severely affected with 16 workers. However, in PhyloMOEA-OMP, the increase speedup decreases slowly. At this point, PhyloMOEA-OMP version have better behavior probably due the increasing communication cost in PhyloMOEA-P.

For some hundred of species, the communication costs become noticeable. This is the case for the *ZILLA\_500* dataset. Figure 16 shows the speedup is affected with more than 8 workers in PhyloMOEA-P and more than 3 workers in PhyloMOEA-OMP.

In practice, it is difficult to determine what is the best combination that optimizes the computation vs. communication ratio in parallel PhyloMOEA. The datasets employed in this experiment were relatively small, mainly due the limited computation resources available. For the largest dataset tested (*ZILLA\_500*), at least, the total execution time can be drastically reduced from 2 days to 6 hours (serial vs. best parallel running times).

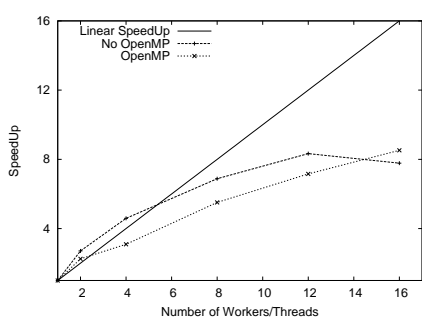


Figure 13: PhyloMOEA-P and PhyloMOEA-OMP speedup for *rbcL\_55* dataset.

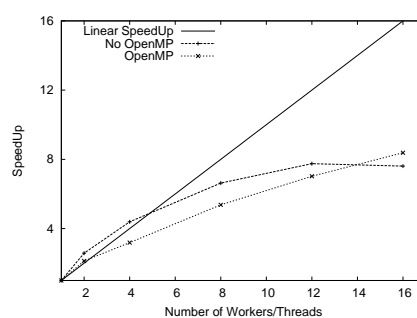


Figure 14: PhyloMOEA-P and PhyloMOEA-OMP speedup for *mtDNA\_186* dataset.

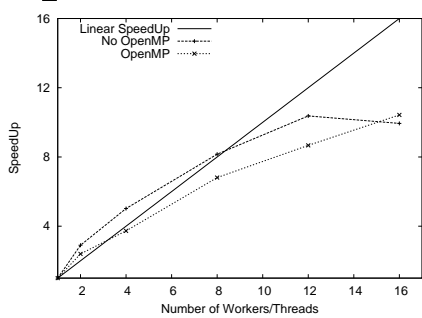


Figure 15: PhyloMOEA-P and PhyloMOEA-OMP speedup for *RDPII\_218* dataset.

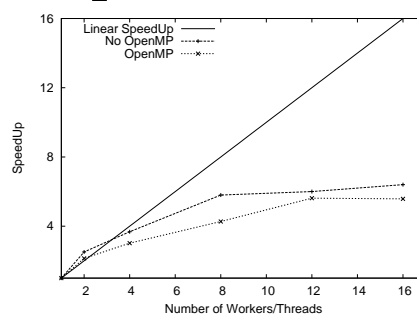


Figure 16: PhyloMOEA-P and PhyloMOEA-OMP speedup for *ZILLA\_500* dataset.

## 7 Conclusion and future challenges

The PhyloMOEA development was motivated by several studies in the literature [27, 30, 33, 56], which point out that various phylogenetic inference methods lead to inconsistent solutions. Previous experiment using PhyloMOEA shows that parsimony and likelihood criteria yield to different trees when they are applied separately [4]. On the other hand, the two main PhyloMOEA modules (MOEA and phylogenetic) were developed from scratch without using mature ready to use metaheuristics and phylogenetic frameworks. This factor limits the incorporation of new features to the program.

This report describes PhyloMOEA porting efforts to the ParadisEO metaheuristic framework developed by INRIA Dolphin team. This framework provides ready-to-use components which simplified the program effort necessary. Moreover, several contributions were incorporate to ParadisEO, mainly new general purposes classes and bug fixing. On the other hand, the Bio++ bioinformatic framework [14] contains phylogenetic inference components more complete and mature than the PhyloMOEA ones. The PhyloMOEA port to Bio++ components was begun although it was not finished yet.

The employ of ParadisEO framework allows new PhyloMOEA to maintain all feature of the previous version. Besides, the ParadisEO-PEO component provides to PhyloMOEA new distributed and parallel capabilities. Two metaheuristics parallel levels were explored in PhyloMOEA:

- PhyloMOEA-P, which distributes the evaluation functions (parsimony and likelihood) across the several workers.
- PhyloMOEA-OMP, focused on parallelizing the objective functions using a multi-thread approach provided by OpenMP.

Results from both versions using several benchmark DNA datasets show sub-linear speedup in most of the cases. Nevertheless, the execution time reduction compared to the serial version is significant. Moreover, the evaluation the optimality criteria (mainly the likelihood function) is considered as the major bottleneck of phylogenetic inference programs. In this regard, the distributed/parallel objective function evaluation implemented in PhyloMOEA-P and PhyloMOEA-OMP are the most straightforward solution for this problem.

On the other hand, memory requirements for evaluating phylogenetic trees with very big datasets easily surpass the capacity of a single machines. In this case, data must be distributed across several nodes in order to evaluate a single tree [47]. Thus, the data partitioning approach is a feature that PhyloMOEA should incorporate in a future version.

Regarding PhyloMOEA parallelization at algorithmic level, several cooperative PhyloMOEA heuristics can explore diverse regions of search space. Besides, more sophisticated (yet time-consuming) recombination and mutation operators can be implemented. Current PhyloMOEA genetic operators are pretty simple, specially in the case of the recombination, and tend to be very disruptive. New heuristics that not only deals with topological modifications, but also take account of branch length and sequence evolutionary parameter optimization are necessary.

To sum up, the PhyloMOEA new version developed using the Paradise-EO framework is feature par with previous one. The ParadisEO focus on reusable

component design accelerate coding and minimizes program efforts. Additionally, the ParadisEO-PEO module allows to easily incorporate parallelism to PhyloMOEA at iteration and solution levels. The new parallel PhyloMOEA programs, namely PhyloMOEA-P and PhyloMOEA-OMP, reduces significantly the execution time required compared to the serial version. However, it is important to point out that these improvements do not modify the algorithm internal design. The current PhyloMOEA issues mentioned above will be addressed in future research.

## Acknowledgment

Experiments presented in this report were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see <https://www.grid5000.fr>).

## References

- [1] D. Bader, U. Roshan, and A. Stamatakis. Computational Grand Challenges in Assembling the Tree of Life: Problems and Solutions. *Advances in Computers*, 68:128, 2006.
- [2] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA — a platform and programming language independent interface for search algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science, pages 494 – 508, Berlin, 2003. Springer.
- [3] S. Cahon, N. Melab, and E. Talbi. Paradiseo: a framework for the flexible design of parallel and distributed hybrid metaheuristics. *Journal of Heuristics*, 10:357–380, 2004.
- [4] W. Cancino and A. Delbem. A multi-objective evolutionary approach for phylogenetic inference. In *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 428–442. Springer Berlin / Heidelberg, 2007.
- [5] W. Cancino and A. Delbem. Inferring phylogenies by multi-objective evolutionary algorithms. *International Journal of Information Technology and Intelligent Computing*, 2(2), 2007.
- [6] L. Cavalli-Sforza and A. Edwards. Phylogenetic Analysis: Models and Estimation Procedures. *Evolution*, 21(3):550–570, 1967.
- [7] C. Ceron, J. Dopazo, E. Zapata, J. Carazo, and O. Trelles. Parallel implementation of DNAm1 program on message-passing architectures. *Parallel Computing*, 24:701–716, JUN 1998.
- [8] M. Charalambous, P. Trancoso, and A. Stamatakis. Initial Experiences Porting a Bioinformatics Application to a Graphics Processor. In *Lecture Notes in Computer Science*, volume 3746, page 415. Springer, 2005.

- [9] C. Coarfa, Y. Dotsenko, J. Mellor-Crummey, L. Nakhleh, and U. Roshan. Prec-i-dcm3: A parallel framework for fast and accurate large scale phylogeny reconstruction. In *11th International Conference on Parallel and Distributed System*, volume 2, pages 346–350, 2005.
- [10] G. Coelho, A. Silva, and F. von Zuben. A multiobjective approach to phylogenetic trees: Selecting the most promising solutions from the pareto front. In *7th International Conference on Intelligent Systems Design and Applications*, 2007.
- [11] C. Coello. The EMOO repository: a resource for doing research in evolutionary multiobjective optimization. *IEEE Computational Intelligence Magazine*, 1:37 – 45, Feb. 2006.
- [12] J. Cole, B. Chai, R. Farris, Wang, S. Kulam, D. McGarrell, G. Garrity, and J. Tiedje. The Ribosomal Database Project (RDP-II): Sequences and Tools for High-throughput rRNA Analysis. *Nucleic Acids Research*, 33: D294–D296, 2005.
- [13] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.
- [14] J. Dutheil, S. Gaillard, E. Bazin, S. Glemin, V. Ranwez, N. Galtier, and K. Belkhir. Bio++: a set of c++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics. *BMC Bioinformatics*, 7:–, 4 2006. doi: 10.1186/1471-2105-7-188.
- [15] J. Felsenstein. Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
- [16] J. Felsenstein. Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution*, 39(4):783–791, 1985.
- [17] J. Felsenstein. *Inferring Phylogenies*. Sinauer, Sunderland, Massachusetts, 2004.
- [18] J. Felsenstein. The newick tree format, 2000. URL <http://evolution.genetics.washington.edu/phylip/newicktree.html>.
- [19] J. Felsenstein. PHYLIP (Phylogeny Inference Package), 2000. URL <http://evolution.genetics.washington.edu/phylip.html>.
- [20] W. Fitch. Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology. *Systematic Zoology*, 20(4):406–416, 1972.
- [21] W. Fitch. A non-sequential method for constructing trees and hierarchical classifications. *Journal of Molecular Evolution*, 4(18):30–37., 1981.
- [22] M. Forster, A. Pick, M. Raitner, and C. Bachmaier. *GTL - Graph Template Library Documentation*. University of Pasdau, 2004. URL <http://infosun.fmi.uni-passau.de/GTL/>.

- [23] P. Goloboff. Methods for faster parsimony analysis. *Cladistics*, 12(3):199–220, 1996.
- [24] S. Guindon. *Méthodes et algorithmes pour l'approche statistique en phylogénie*. PhD thesis, U.F.R. Sciences de Montpellier. Université de Montpellier II, 2003.
- [25] S. Guindon and O. Gascuel. A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Systematic Biology*, 5(52):696–704, 2003.
- [26] J. Handl, D. Kell, and J. Knowles. Multiobjective Optimization in Computational Biology and Bioinformatics. *IEEE Transactions on Computational Biology and Bioinformatics*, 4(2):289–292, 2006.
- [27] J. Huelsenbeck. Performance of Phylogenetic Methods in Simulation. *Systematic Biology*, 44:17–48, 1995.
- [28] D. Huson, S. Nettles, and T. Warnow. Disk-covering, a fast converging method for phylogenetic tree reconstruction. *Journal of Computational Biology*, 6(3):369–386, 1999.
- [29] M. Ingman and U. Gyllensten. mtDB: Human Mitochondrial Genome Database, a Resource for Population Genetics and Medical Sciences. *Nucleic Acids Research*, 34:D749–D751, 2006.
- [30] L. Jin and M. Nei. Limitations of the Evolutionary Parsimony Method of Phylogenetic Analysis. *Molecular Biology and Evolution*, 7:82–102, 1990.
- [31] K. Katoh, K. Kuma, and T. Miyata. Genetic Algorithm-Based Maximum-Likelihood Analysis for Molecular Phylogeny. *Journal of Molecular Evolution*, 53:477–484, 2001.
- [32] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, feb 2006.
- [33] M. Kuhner and J. Felsenstein. A Simulation Comparison of Phylogeny Algorithms under Equal and Unequal Evolutionary Rate. *Molecular Biology and Evolution*, 11:459–468, 1994.
- [34] A. R. Lemmon and M. C. Milinkovitch. The Metapopulation Genetic Algorithm: An Efficient Solution for the Problem of Large Phylogeny Estimation. In *Proceedings of the National Academy of Sciences*, volume 99, pages 10516–10521, 2002.
- [35] P. O. Lewis. A Genetic Algorithm for Maximum-Likelihood Phylogeny Inference Using Nucleotide Sequence Data. *Molecular Biology and Evolution*, 15(3):277–283, 1998.
- [36] C. Michener and R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.

- [37] B. Minh, L. Vinh, A. von Haeseler, and H. Schmidt. pIQPNNI: parallel reconstruction of large maximum likelihood phylogenies. *Bioinformatics*, 21(19):3794–3796, 2005.
- [38] B. Minh, L. Vinh, H. Schmidt, and A. Haeseler. Large maximum likelihood trees. In *Proceeding of the NIC Symposium*, volume 32 of *NIC Series*, pages 357–365, 2006.
- [39] G. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek. fastDNAm1: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *Computer Applications in the Biosciences*, 10(1):41–48, 1994.
- [40] L. Poladian and L. Jermiin. Multi-Objective Evolutionary Algorithms and Phylogenetic Inference with Multiple Data Sets. *Soft Computing*, 10(4):359–368, 2006.
- [41] A. Rokas, B. Williams, N. King, and S. Carroll. Genome-Scale Approaches to Resolving Incongruence in Molecular Phylogenies. *Nature*, 425(23):798–804, 2003.
- [42] F. Ronquist, J. Huelsenbeck, and P. van der Mark. *MrBayes 3.1 Manual*. School of Computer Science. Florida State University, 2005.
- [43] N. Saitou and M. Nei. The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.
- [44] D. Sankoff. Simultaneous Solution of the RNA Folding, Alignment and Proto-Sequence Problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825, 1985.
- [45] A. Stamatakis. Raxml-vi-hpc: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, NOV 1 2006. doi: 10.1093/bioinformatics/btl446.
- [46] A. Stamatakis and H. Meier. New Fast and Accurate Heuristics for Inference of Large Phylogenetic Trees. In *18th IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS2004)*, 2004.
- [47] A. Stamatakis and M. Ott. Exploiting Fine-Grained Parallelism in the Phylogenetic Likelihood Function with MPI, Pthreads, and OpenMP: A Performance Study. In *Proceedings of the Third IAPR International Conference on Pattern Recognition in Bioinformatics*, pages 424–435. Springer, 2008.
- [48] A. Stamatakis, T. Ludwig, and H. Meier. Parallel inference of a 10.000-taxon phylogeny with maximum likelihood. In *Proceeding of the Euro-Par2004*, Lecture Notes in Computer Science, pages 997–1004. Springer, 2004.
- [49] A. Stamatakis, T. Ludwig, and H. Meier. Raxml-ii: a program for sequential, parallel and distributed inference of large phylogenetic. *Concurrency and Computation-Practice & Experience*, 17(14):1705–1723, Sp. Iss. SI DEC 2005.

- [50] A. Stamatakis, M. Ott, and T. Ludwig. RAxML-OMP: An Efficient Program for Phylogenetic Inference on SMPs. In *Lecture Notes in Computer Science*, volume 3606, page 288. Springer, 2005.
- [51] A. Stamatakis, F. Blagojevic, D. Nikolopoulos, and C. Antonopoulos. Exploring new search algorithms and hardware for phylogenetics: RAxML meets the IBM cell. *The Journal of VLSI Signal Processing*, 48(3):271–286, 2007.
- [52] C. Stewart, D. Hart, D. Berry, G. Olsen, E. Wernert, and W. Fischer. Parallel implementation and performance of fastDNAm1-a program for maximum likelihood phylogenetic inference. In *Proceedings of SC2001*, volume 302, 2001.
- [53] K. Strimmer and A. von Haesler. Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution*, 13:407–514, 1996.
- [54] D. Swofford, G. Olsen, P. Waddell, and D. Hillis. Phylogeny Reconstruction. In *Molecular Systematics*, chapter 11, pages 407–514. Sinauer, 3 edition, 1996.
- [55] E. Talbi. *Metaheuristics: from design to implementation*. Wiley, 2009.
- [56] Y. Tatenno, N. Takezaki, and M. Nei. Relative Efficiencies of the Maximum-Likelihood, Neighbor-Joining, and Maximum Parsimony Methods when Substitution Rate Varies with Site. *Molecular Biology and Evolution*, 11: 261–267, 1994.
- [57] L. Vinh and A. von Haeseler. Iqpnni: Moving fast through tree space and stopping in time. *Molecular Biology and Evolution*, 21(8):1565–1571, AUG 2004.
- [58] Z. Yang. Maximum Likelihood Estimation on Large Phylogenies and Analysis of Adaptative Evolution in Human Influenza Virus A. *Journal of Molecular Evolution*, 51(5):423–432, 2000.
- [59] Z. Yang. Maximum-Likelihood Estimation of Phylogeny from DNA Sequences when Substitution Rates Differ over Sites. *Molecular Biology and Evolution*, 10(6):1396–1401, 1993.
- [60] Z. Yang. Maximum-likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *Journal of Molecular evolution*, 39(3):306–314, 1994.
- [61] Z. Yang. *Computational molecular evolution*. Oxford series in ecology and evolution. Oxford University Press, Oxford, 2006.
- [62] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In X. Yao et al., editors, *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer, 2004.

- 
- [63] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.
- [64] D. Zwickl. *Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets under the Maximum Likelihood Criterion*. PhD thesis, Faculty of the Graduate School. University of Texas., 2006.

## A Comparison of alternative MOEA models in PhyloMOEA

In this appendix, results from NSGA-II and IBEA from experiments in Section 6 are compared. The following steps were performed based on guidelines described in [32]:

- The upper and lower bound for each objective function were calculated. This procedure involves collecting all Pareto-fronts obtained in each execution for NSGA-II and IBEA.
- Using the bound obtained previously, all objective function scores were normalized.
- A reference set is obtained by calculating the non-dominated solutions from the all the normalized solutions.
- Hypervolume and Epsilon unary indicators [32] (denoted by  $I_H^-$  and  $I_{\epsilon+}^1$ , respectively) were obtained from the normalized Pareto-front and the reference set.
- The Fisher matched sample test [32] was performed independent on  $I_H^-$  and  $I_{\epsilon+}^1$  indicators. This test was selected due that in all executions, the same initial population was maintained for both NSGA-II and IBEA.

All the statistical test were carry out using the PISA framework [2]. Tables 2 and 3 shows the  $P$ -values obtained by the Fisher matched test in each dataset for the the  $I_H^-$  and  $I_{\epsilon+}^1$  unary indicators, respectively. The bold numbers indicate  $P$ -values that show significantly differences between IBEA and NSGA-II. In the *rbcL\_55* and *RDPII\_218* datasets, IBEA performs better than NSGA-II according the  $I_H^-$  and  $I_{\epsilon+}^1$  indicators while NSGA-II obtain the best results in the *mtDNA\_186* dataset. In the *ZILLA\_500* dataset, no significantly differences were found. The mixed results shows no clear superiority of IBEA or NSGA-II.

Table 2: Results from the Fischer matched sample test using the  $I_H^-$  indicator

	<i>rbcL_55</i>	<i>mtDNA_186</i>	<i>RDPII_218</i>	<i>ZILLA_500</i>
IBEA	<b>0.0137</b>	0.9878	<b>0.0091</b>	0.4891
NSGA-II	0.9862	<b>0.0105</b>	0.9898	0.5062

Table 3: Results from the Fischer matched sample test using the  $I_{\epsilon+}^1$  indicator.

p-values	<i>rbcL_55</i>	<i>mtDNA_186</i>	<i>RDPII_218</i>	<i>ZILLA_500</i>
IBEA	<b>0.0078</b>	0.9429	<b>0.0306</b>	0.4581
NSGA-II	1.0000	<b>0.0525</b>	0.9684	0.5383



---

Centre de recherche INRIA Lille – Nord Europe  
Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex  
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier  
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique  
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex

Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex  
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex  
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex  
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

---

Éditeur  
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)  
<http://www.inria.fr>  
ISSN 0249-6399