Multiobjective Frog-Leaping Optimization for the Study of Ancestral Relationships in Protein Data

Sergio Santander-Jiménez, Miguel A. Vega-Rodríguez, and Leonel Sousa

***Abstract*—Among the different scientific domains where metaheuristics find applicability, bioinformatics represents a particularly challenging field due to the multiple complexity factors involved in the processing of biological data. In this context, the exploration of protein sequence data is remarkably increasing the temporal demands of such biological problems, thus motivating the interest in investigating new approaches that effectively combine bioinspired metaheuristics and parallelism. This work addresses the reconstruction of ancestral relationships from amino acid sequences by using a multiobjective approach based on the shuffled frog-leaping optimization technique. Due to the inherent parallel nature of this approach, we define different parallel schemes aimed at exploiting the computing capabilities of modern cluster platforms. The experiments performed in five real datasets give account of the relevance of using parallelism-aware metaheuristic designs, as well as the need to consider both parallel performance and solution quality when tackling such difficult optimization scenarios.**

***Index Terms*—Bioinspired computing, parallelism, multiobjective optimization, bioinformatics.**

I. Introduction

THROUGHOUT the years, the design of bioinspired metaheuristics has been attracting major research interest as an alternative to traditional optimization techniques to deal with NP-hard problems. These problems are characterized by the need to explore a decision space S, seeking for those solutions that optimize an objective function $f : S \rightarrow \mathbb{R}$ (or multiple ones in the case of multiobjective optimization problems). When tackling such problems, the optimization process becomes a challenging issue due to the presence of exponentially growing decision spaces and time-consuming objective functions. In this sense, the advances in metaheuristic search engines, along with the possibility of integrating parallel computing techniques into them, have given rise to robust algorithmic approaches which have been successfully applied in a wide range of scientific domains [1].

However, the introduction of more realistic assumptions and the publication of computationally demanding datasets have significantly increased the complexity associated to optimization problems in real-world contexts. Computational biology is a representative example of a research field in which the problems to be tackled stand out due to their hardness. Although the processing of large volumes of biological data represents a grand computational challenge by itself, nowadays there is special emphasis on the study of problems based on protein sequence data [2]. This represents a further step in terms of complexity with regard to traditional DNA-based analyses, since the number of possible character states grows from 4 nucleotides to 20 amino acids thus having a noticeable impact e.g., in the objective function calculations.

This work is focused on tackling a well-known bioinformatics problem, the reconstruction of evolutionary relationships among organisms considering protein-based scenarios [3]. In this type of biological analyses, the processing of protein sequence data is said to provide useful knowledge e.g., for the gene annotation and discovery tasks, the prediction of gene function, and the construction of gene families [4]. As such complex data is increasingly playing a major role in current phylogenetic studies [5], new computational approaches combining stochastic procedures and parallelism are required to provide high-quality solutions in reasonable time.

In order to address this problem, we undertake the study of a multiobjective bioinspired approach based on the metaheuristic Shuffled Frog-Leaping Algorithm (SFLA) [6], [7]. The proposed design takes the basic features of this algorithm (swarm techniques and parallel searches) and adapts them to a multiobjective formulation of the problem. In addition, given the intrinsically parallel nature of its metaheuristic design, we define different parallel schemes to allow the exploitation of high-performance computing systems using the MPI and OpenMP standards [8]. By performing experimentation over real-world amino acid datasets, we will carry out the evaluation of the designed approach from the perspectives of parallel performance and solution quality, assessing our results by means of comparisons with other multiobjective approaches and biological methods from the state of the art.

This paper is organized as follows. The next section summarizes representative works related to the study of protein data for the targeted problem, whose formulation is explained in Section III. Section IV describes the main features of the proposed approach and its parallelization under MPI+OpenMP approaches. Section V reports experiments and evaluates results by adopting different performance metrics. Finally, conclusions and future work lines are included in Section VI.

II. Related Work

The reconstruction of evolutionary relationships has been tackled in the literature from different perspectives, with a significant amount of works based on bioinspired computing approaches. Comprehensive surveys on this subject (with

Sergio Santander-Jiménez and Leonel Sousa are with the INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa 1000-029, Portugal (e-mail: sergio.jimenez@tecnico.ulisboa.pt, leonel.sousa@ist.utl.pt).

Miguel A. Vega-Rodríguez is with the Department of Computer and Communications Technologies, University of Extremadura, Escuela Politécnica, Campus Universitario s/n, Caceres 10003, Spain (e-mail: mavega@unex.es)

special focus on DNA approaches) can be found in [9], [10], [11]. This section will highlight the most relevant proposals to address this problem in the protein domain.

Early works on this topic pointed out the need to combine different techniques to deal with the computational challenge that this problem represents even on small-sized datasets. Matsuda et al. [12] provided insight into the additional complexity associated to the analysis of protein sequence data and the need to apply parallelism and stochastic methods to reduce execution time. That study led to the proposal of the first genetic algorithm for maximum likelihood phylogenetic reconstruction [13], analyzing 17 EF-1α protein sequences for experimental purposes. Later on, Reijmers et al. reported in [14] a genetic algorithm with distance-based solution encoding to analyze amino acid sequences of 37 G protein-coupled receptors (GPCRs). The combination of high-performance computing and evolutionary computation to address the problem was undertaken by Katoh et al. [15], who proposed a parallel genetic algorithm for processing amino acid datasets enclosing up to 24 taxa. Pond and Frost addressed the evolutionary model selection issue at the protein level by means of an MPI-based parallel genetic algorithm [16]. In addition, Hill et al. proposed a genetic algorithm to perform phylogenetic analyses on amino acid sequences under the maximum parsimony criterion [17], conducting the validation of the proposal over datasets of human membrane-bound GPCRs. Other related problems, such as the inference of protein-protein functional interactions from phylogenetic profiles, have also been tackled by using these techniques [18].

Due to the publication of increasing volumes of biological data, more recent works have been aimed at solving the deficiencies shown by previous approaches in terms of both processing time and biological quality. As an illustration, the GARLI software proposed by Zwickl [19] addresses these issues by hybridizing a genetic algorithm with efficient local search operators. This approach has been extended by integrating parallelism to conduct high-performance and high-throughput analyses at the nucleotide and amino acid levels [20]. On the other hand, MetaPIGA [21] comprises different metaheuristics (including simulated annealing, a traditional genetic algorithm, and a metapopulation genetic algorithm) supported on the Java Multi-Threading technology to take advantage of shared-memory hardware setups. A simulated annealing search algorithm with support for protein models was reported by Stamatakis in [22] for the RAxML tool, one of the current reference methods to perform large-scale phylogenetic analyses adopting the likelihood criterion [23]. Furthermore, other state-of-the-art approaches, such as TNT [24] (maximum parsimony), IQ-TREE [25] (maximum likelihood), and MrBayes [26] (Bayesian inference), also allow the high-performance parallel processing of protein data.

Regarding multiobjective approaches, different proposals have been reported to carry out the inference of ancestral relationships over conflicting datasets [27] and divergent optimality criteria [28]. We can highlight in this context two relevant proposals due to Coelho et al. [29], who reported an immune-inspired approach combining different distance-based metrics, and Cancino and Delbem [30], who proposed the NSGA-II-based tool PhyloMOEA to conduct inferences under parsimony and likelihood. Nevertheless, to the best of our knowledge all of these multiobjective approaches are focused on the analysis of nucleotide datasets without considering their evaluation on protein-based scenarios. The only study close to the protein domain was reported by Jayaswal et al. [31], who processed protein-coding mitochondrial DNA sequences by taking into account divergence on evolution patterns at the first and second codon sites. In the work herein presented, we aim to go a step further and address the protein-based reconstruction challenge by designing a novel multiobjective approach based on swarm techniques and parallelism awareness. Hence, the main contributions of this work can be summarized as:

- Proposal of a multiobjective approach based on the parallelism-aware metaheuristic SFLA, adapting it to the inference of evolutionary relationships as a case study;
- Identification of parallel opportunities and challenges in the algorithmic design to define different parallel schemes (including a novel proposal based on the use of trial counters to improve load balancing);
- Evaluation of the proposal over five real-world amino acid datasets, undertaking 1) a thorough analysis of parallel scalability to determine the efficiency of the proposed schemes, and 2) the parametric study of the proposal considering both multiobjective and parallel performance;
- Solution quality assessment through comparisons with up to 8 reference methods (Non-Dominated Sorting Genetic Algorithm II NSGA-II, the Indicator-Based Evolutionary Algorithm IBEA, and 6 state-of-the-art biological tools).

III. Problem Formulation

The phylogeny reconstruction problem is aimed at describing evolutionary hypotheses by inferring ancestral relationships from biological data [3]. Let N be the number of species to be studied and M the length of their aligned sequences, defined under the amino acid character state alphabet Λ in protein analyses. The observed divergence and similarities allows the description of evolutionary relationships through phylogenies. In a phylogeny $T = (V, E)$, the node set V contains the species characterized in the input data (terminal nodes) and hypothetical ancestors (internal nodes), while the branch set E defines ancestral linkages between nodes.

This optimization problem involves the exploration of the phylogeny search space to find the solution which maximizes or minimizes a biological criterion. Taking into account the benefits that imply the simultaneous use of different criteria to address incongruence issues [28], [29], [30], we consider a multiobjective formulation based on two of the most widely-used phylogenetic objective functions: parsimony $P(T)$ and likelihood $L(T)$. The key idea consists of finding a satisfying approximation to the set of solutions which are not dominated[1] by any other one in the solution space, that is, the Pareto-optimal set containing the best tradeoffs among the objectives.

[1]Given two solutions $x, y \in S$ to a multiobjective optimization problem with k objectives, we say that x dominates y ($x \succ y$) iff $\forall\ i \in [1, 2, ..., k]$, $f_i(x)$ is not worse than $f_i(y)$ and $\exists\ i \in [1, 2, ..., k]$ such that $f_i(x)$ is better than $f_i(y)$. If the second condition is not verified but x is still not worse than y in all the objectives, x is said to weakly dominate y ($x \succeq y$).

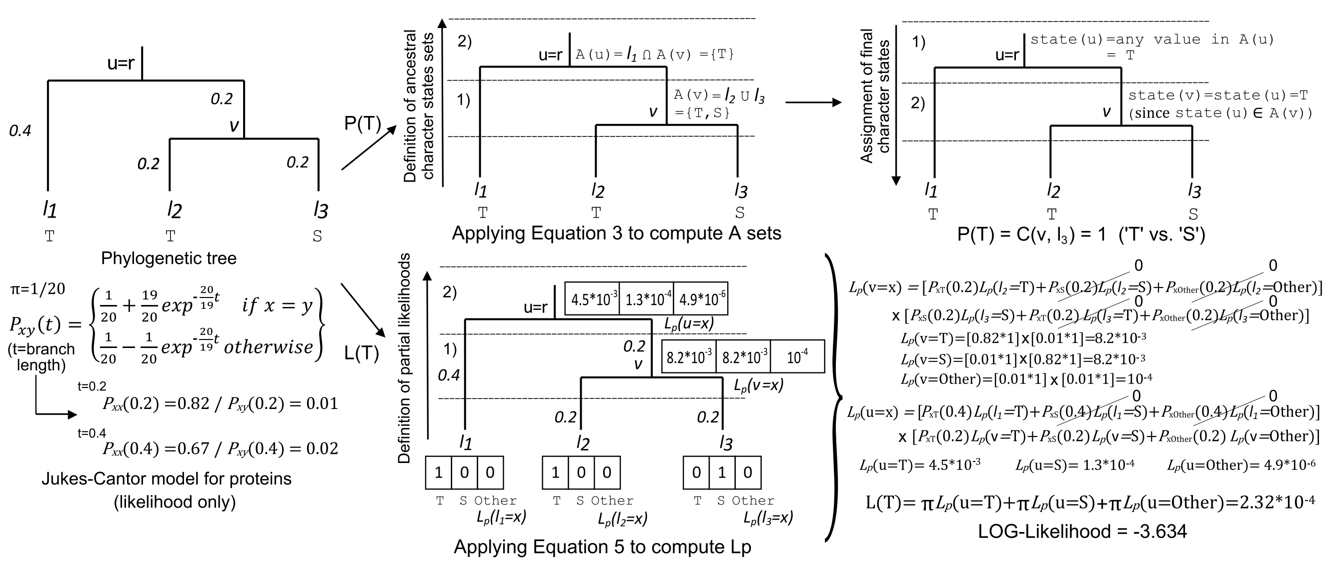

u=r
0.4
0.2
v
0.2
0.2
l1 T
l2 T
l3 S
Phylogenetic tree
P(T)
Definition of ancestral character states sets
2) u=r A(u) = l1 ∩ A(v) = {T}
1) A(v) = l2 ∪ l3 = {T,S}
Applying Equation 3 to compute A sets
Assignment of final character states
1) state(u)=any value in A(u) = T
2) state(v)=state(u)=T (since state(u) ∈ A(v))
P(T) = C(v, l3) = 1 ('T' vs. 'S')
π=1/20
Pxy(t) = 1/20 + 19/20 exp^(-20/19 t) if x = y; 1/20 − 1/20 exp^(-20/19 t) otherwise
(t=branch length)
t=0.2 Pxx(0.2) = 0.82 / Pxy(0.2) = 0.01
t=0.4 Pxx(0.4) = 0.67 / Pxy(0.4) = 0.02
Jukes-Cantor model for proteins (likelihood only)
L(T)
Definition of partial likelihoods
2) u=r 4.5*10^-3 1.3*10^-4 4.9*10^-6 Lp(u=x)
1) 0.4 0.2 v 8.2*10^-3 8.2*10^-3 10^-4 Lp(v=x)
1 0 0 / 1 0 0 / 0 1 0
T S Other Lp(l1=x) Lp(l2=x) Lp(l3=x)
Applying Equation 5 to compute Lp
Lp(v=x) = [PxT(0.2)Lp(l2=T)+PxS(0.2)Lp(l2=S)+PxOther(0.2)Lp(l2=Other)] x [PxS(0.2)Lp(l3=S)+PxT(0.2)Lp(l3=T)+PxOther(0.2)Lp(l3=Other)]
Lp(v=T)=[0.82*1]x[0.01*1]=8.2*10^-3
Lp(v=S)=[0.01*1]x[0.82*1]=8.2*10^-3
Lp(v=Other)=[0.01*1] x [0.01*1]=10^-4
Lp(u=x) = [PxT(0.4)Lp(l1=T)+PxS(0.4)Lp(l1=S)+PxOther(0.4)Lp(l1=Other)] x [PxT(0.2)Lp(v=T)+PxS(0.2)Lp(v=S)+PxOther(0.2)Lp(v=Other)]
Lp(u=T)= 4.5*10^-3 Lp(u=S)= 1.3*10^-4 Lp(u=Other)= 4.9*10^-6
L(T)= πLp(u=T)+πLp(u=S)+πLp(u=Other)=2.32*10^-4
LOG-Likelihood = -3.634

Fig. 1. Example of calculation of parsimony P(T) and likelihood L(T) for a single character. In the likelihood case, we assume for the sake of simplicity that any amino acid different from 'T' (Threonine) and 'S' (Serine) is represented by the 'Other' state

A case study on the Plethodontid salamanders phylogeny [32] reported in [33] gives account of the relevance of applying multiobjective approaches to recover accurate solutions in conflicting phylogenetic scenarios, providing compromise answers jointly supported by the different objectives under consideration. Beyond that, multiobjective approaches are also able to provide insight into the relationship between the characteristics of the input data and the observed phylogenetic conflicts, giving researchers more decision-making opportunities due to the multiple Pareto solutions that are inferred while also maintaining significant solution quality attending to each objective separately [29], [33]. Formally, the multiobjective formulation of the problem can be expressed as:

$$\begin{aligned} \text{optimize } \vec{f}(T) &= \{f_1(T), f_2(T)\}, \\ \text{where } f_1(T) &= \text{minimize } P(T) \in \mathbb{Z}, \\ f_2(T) &= \text{maximize } L(T) \in \mathbb{R}. \end{aligned} \tag{1}$$

On the one hand, the parsimony objective aims to minimize the amount of evolutionary change observed between related nodes in the phylogenetic topology:

$$P(T) = \sum_{i=1}^{M} \sum_{(u,v)\in E} C(u_i, v_i), \tag{2}$$

where $(u, v) \in E$ is the branch which associates two nodes $u, v \in V$, $u_i, v_i \in \Lambda$ the character states at the ith site of the sequences for u and v, and $C(u_i, v_i)$ measures if a substitution event has taken place ($C(u_i, v_i) = 1$) or not ($C(u_i, v_i) = 0$) between u_i and v_i, that is, if a change has been observed between u and v states at the ith site. While the input sequences define the character states for the terminal nodes, the character states for the internal nodes can be computed through the application of Fitch's algorithm [3]. In a first step, the phylogenetic tree to be evaluated is processed in a bottom-up fashion, assigning to each internal node a set of possible ancestral states A_i at the ith site. Given an internal node u with children v, w, $A_i(u)$ is calculated as:

$$A_i(u) = \begin{cases} A_i(v) \cap A_i(w) & \text{if } A_i(v) \cap A_i(w) \neq 0, \\ A_i(v) \cup A_i(w) & \text{if } A_i(v) \cap A_i(w) = 0. \end{cases} \tag{3}$$

Once defined the A_i sets, we compute the final character states by processing the phylogenetic tree in a top-down fashion. Starting from the root r, a randomly chosen state in $A_i(r)$ is selected as its final character state r_i. For any remaining internal node u with ancestor h, u_i takes the value h_i iff h_i is one of the states included in $A_i(u)$. Otherwise, a random state among the ones in $A_i(u)$ is chosen as u_i.

On the other hand, the likelihood objective seeks to maximize the conditional probability of observing the character states in the input sequences given a phylogenetic tree topology and a probabilistic model of sequence evolution:

$$L(T) = \prod_{i=1}^{M} \sum_{x\in\Lambda} \pi_x L_p(r_i = x), \tag{4}$$

where π_x is the stationary probability of the character state $x \in \Lambda$ and $L_p(r_i = x)$ the partial likelihood of observing x in the root node $r \in V$ at the ith site. The computation of partial likelihoods can be computed through a recursive approach defined in Felsenstein's algorithm [3]. Given an internal node u with children v, w, we calculate $L_p(u_i = x)$ as follows:

$$L_p(u_i = x) = \left(\sum_{y\in\Lambda} P_{xy}(t_{uv}) L_p(v_i = y)\right) \times \left(\sum_{y\in\Lambda} P_{xy}(t_{uw}) L_p(w_i = y)\right), \tag{5}$$

where t_{uv}, t_{uw} are the evolutionary times (branch length values) between u and v, w and $P_{xy}(t)$ the probability of observing a substitution event from x to y within a time t, which is provided by the model of sequence evolution considered in the likelihood calculations. For a terminal node l, $L_p(l_i = x)$ will be 1 iff $l_i = x$ and 0 otherwise. Once defined the partial likelihoods for each state x, Equation 4 provides the likelihood score of the topology, which is usually reported in terms of log-likelihood values. Figure 1 provides a simple example of parsimony and likelihood calculations.

The analysis of protein sequence data in this context is stated to provide some benefits, e.g., when dealing with large evolutionary distances, since the functional properties defined by the 20 possible amino acids allow a higher resolution than other types of data [3]. However, the consideration of a wider range of character states can affect the time complexity of the problem. More specifically, the temporal costs contributed by the evaluation procedures are generally influenced not only by the length M of the input sequences but also by the number of possible character states in Λ. In addition, this optimization problem requires the processing of huge search spaces which grow exponentially with the number of input sequences N according to the double factorial $(2N-5)!!$ [3], [4]. As a result, it has been demonstrated that phylogeny reconstructions under both parsimony and likelihood represent NP-hard problems [34], [35]. These complexity factors justify the need to explore novel approaches to address this challenging problem.

IV. Multiobjective Frog-Leaping Optimization

In order to tackle phylogeny reconstructions from protein data, we propose a multiobjective approach based on the metaheuristic SFLA. This section details the main features of this method, its adaptation to the problem, and the integration of high-performance computing techniques to take advantage of the intrinsic parallel nature of the algorithmic design.

A. Algorithmic Design

SFLA [6] is a population-based metaheuristic built upon the idea of combining search strategies from Particle Swarm Optimization with the mixing of information from parallel local searches described in the Shuffled Complex Evolution technique. Modelled after the intra and inter-group interactions between frog communities in a swamp, this algorithm addresses optimization problems by processing different partitions of solutions in parallel, which are eventually combined to allow a global exchange of information among individuals.

At each generation, the population in SFLA is partitioned into subsets of individuals, named memeplexes, which learn separately for a certain number of learning steps with the aim of exploring different directions of the search space. New solutions are generated through the sharing of information at the memeplex level by taking as reference the best local solution in the corresponding partition. This mechanism can be complemented by considering also the best global individual identified in the population, in accordance with the status of the optimization process. Once each memeplex has been processed, the algorithm carries out a shuffling step to update the memeplexes in such a way that the best individuals are equally distributed among the partitions to boost the evolution

Algorithm 1 Multiobjective Shuffled Frog-Leaping Algorithm

```
Input: maxEval (maximum number of evaluations), popSize (number of individuals
    in the population), $m$ (number of memeplexes), $n$ (number of individuals per
    memeplex = popSize/m), $n_l$ (number of learning steps per memeplex).
Output: PF (non-dominated solutions found by the algorithm).
 1: $P \leftarrow$ Initialize Population ($P$, popSize)
 2: $PF \leftarrow 0$
 3: while ! stop criterion is reached (maxEval) do
 4:     $P \leftarrow$ Fast Non-Dominated Sort and Crowding Computation ($P$, popSize)
 5:     $\{Mem_1 \ldots Mem_m\} \leftarrow$ Shuffle and Distribute into Memeplexes ($P$, popSize)
 6:     $Best_{global} \leftarrow$ Identify Best Global ($\{Mem_1 \ldots Mem_m\}$)
 7:     for $i$ = 1 to $m$ do
 8:         $Best_{local} \leftarrow$ Identify Best Local ($Mem_i$)
 9:         for $j$ = 1 to $n_l$ do
10:             $P'_{new} \leftarrow$ Learn from Best Local ($Best_{local}$, $Mem_{i(n-j)}$)
11:             if ! $P'_{new} \succ Mem_{i(n-j)}$ then
12:                 $P'_{new} \leftarrow$ Learn from Best Global ($Best_{global}$, $Mem_{i(n-j)}$)
13:                 if ! $P'_{new} \succ Mem_{i(n-j)}$ then
14:                     $P'_{new} \leftarrow$ Apply Local Search ($Mem_{i(n-j)}$)
15:                 end if
16:             end if
17:             $Mem_{i(n-j)} \leftarrow P'_{new}$
18:         end for
19:     end for
20:     $P \leftarrow$ Merge Memeplexes ($P$, $\{Mem_1 \ldots Mem_m\}$)
21:     $PF \leftarrow$ Update Pareto Front ($P$)
22: end while
23: return PF
```

of the overall population. Different examples of successful applications (such as water distribution and power flow optimization, multi-user detection, production planning, etc.) point out the relevance of SFLA to address challenging problems [7]. In fact, SFLA has led to significant results in hard-to-solve bioinformatics problems, including RNA secondary structure prediction [36] and biomedical data feature selection [37].

In this work, we study a multiobjective approach based on SFLA named as Multiobjective Shuffled Frog-Leaping Algorithm (MO-SFLA), which takes advantage of the multiobjective quality mechanisms from NSGA-II [38] in order to adapt the SFLA features to the multiobjective context. Algorithm 1 details the pseudocode of the proposed method. Its application to the problem addressed in this study involves an individual representation based on an indirect encoding of phylogenetic topologies using distance matrices [3]. In this representation, a solution is codified by a $N \times N$ symmetric matrix δ (N being the number of input sequences), with each entry $\delta[x, y]$ representing the evolutionary divergence between two terminal nodes x and y by a floating point number. The use of distance matrices to encode solutions has been reported to lead to significant solution quality in previous works, from both single [39] and multiobjective [28] perspectives. When using this kind of representation, a tree-building method must be applied to allow the mapping of the solution from the distance matrix space to the phylogeny space. Herein, we use a neighbour-joining algorithm known as BIONJ [3] to recover phylogenetic topologies from the processed distance matrices.

When initializing the population (line 1 in Algorithm 1), starter topologies are randomly selected from a repository of 1000 phylogenies generated from bootstrap samples of the input alignment, computing afterwards the associated distance matrices. Given a starter solution $T = (V, E)$, its distance matrix δ is calculated by applying the following expression over each entry $\delta[x, y]$: $\delta[x, y] = \sum_{u,v \in PT_{x,y}} t_{uv}$, where $PT_{x,y} \subset V$ refers to the set of nodes included in the path between x

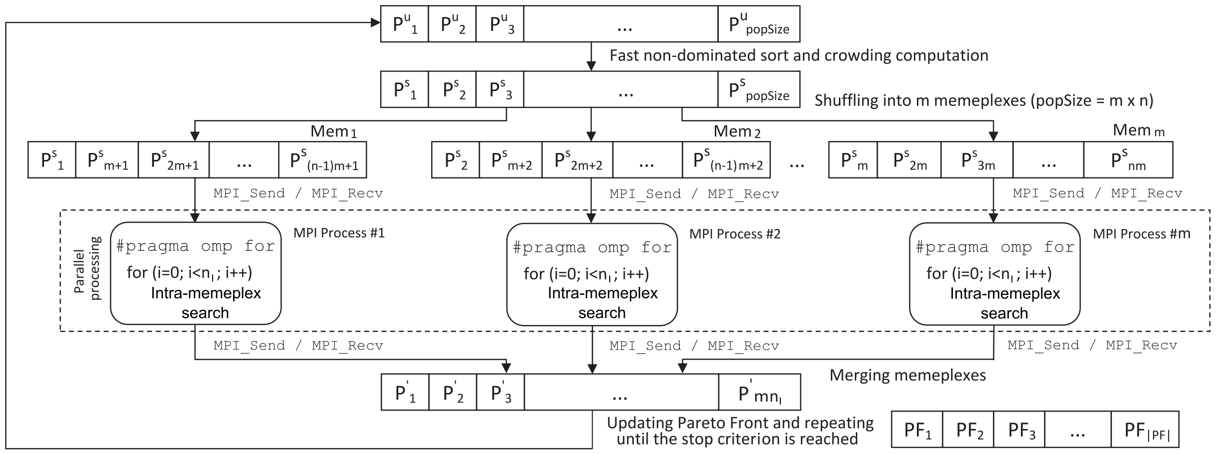


Fig. 2. Coarse-grained representation of the parallel processing of memeplexes in MO-SFLA

and y, while t_{uv} represents the length value of the branch which connects the nodes $u, v \in PT_{x,y}$. Once generated the initial population, we initialize the Pareto front structure *PF* which will be used to store the non-dominated solutions found throughout the execution of the metaheuristic (line 2). Afterwards, MO-SFLA proceeds with its main loop until a stop criterion (e.g., a maximum number of evaluations) is satisfied.

A generation begins by computing Pareto ranks and density information values for each individual in the population, applying to this end a fast non-dominated sort and crowding distance ordering [38] (line 4). These values will be used to identify solution quality, giving preference to solutions with lower rank and higher crowding distance within the same rank. Afterwards, the shuffling procedure (line 5) divides the population into *m* memeplexes, each one composed of *n* individuals in such a way that *m*×*n*=*popSize*. During the shuffling, the first best individual P_1 according to its multiobjective quality (rank and crowding) is assigned to the first memeplex *Mem*$_1$, the second best individual P_2 goes to the second memeplex *Mem*$_2$, the *m*th best individual goes to the *m*th memeplex *Mem*$_m$, the *m+1*th best individual goes to *Mem*$_1$, and so on.

The next section in MO-SFLA involves parallel searches over each memeplex (lines 7-19). Given a memeplex *Mem*$_i$, its processing is carried out as follows. Firstly, a new distance matrix $P'_{new}.\delta$ is generated from an individual *Mem*$_{ij}$ (line 10) by using the information provided by a reference solution *Ref*.δ, given by the best local individual in *Mem*$_i$ (identified in line 8 as the solution belonging to *Mem*$_i$ which shows the best ranking and crowding distance value in that memeplex):

$$D_{xy} = rand() \cdot (Ref.\delta[x,y] - Mem_{ij}.\delta[x,y]), \quad (6)$$

$$P'_{new}.\delta[x,y] = Mem_{ij}.\delta[x,y] + D_{xy}, \quad (7)$$

where *rand()* is a random number from a uniform distribution in the range [0,1]. Equations 6 and 7 are applied over each entry $P'_{new}.\delta[x,y]$ such that $y < x$, ensuring the symmetry of the resulting distance matrix by assigning $P'_{new}.\delta[y,x] = P'_{new}.\delta[x,y]$. Then, the associated phylogeny $P'_{new}.T$ is inferred and its objective functions scores are calculated. The resulting individual P'_{new} is then compared with *Mem*$_{ij}$ under Pareto dominance. In case P'_{new} does not dominate *Mem*$_{ij}$, a new solution is generated (line 12) by repeating the previous steps using as reference the best global individual in the overall population (identified in line 6 by randomly choosing an individual among the N_{best} ones with better ranking and crowding distance values in order to provide more variability to the multiobjective search, being N_{best} experimentally set to 3) instead of the best local one. If P'_{new} does not dominate after that, this solution is dismissed and replaced by the results of applying over *Mem*$_{ij}$ a phylogeny optimization procedure (local search, line 14) involving the application of subtree pruning-regrafting (SPR) and nearest neighbour interchange (NNI) topological rearrangements. NNI takes a branch of the tree and performs a swap between the subtrees at the sides of that branch, while SPR removes a subtree from the phylogenetic topology and regrafts it in a different place. These operators are applied following the Progressive Tree Neighborhood methodology [40], using gradient descent [3] to optimize the branch lengths of the resulting phylogeny. The individual obtained by this last procedure is accepted regardless of the dominance checking in order to improve the diversity of solutions managed by the search engine.

After n_l iterations over *Mem*$_i$, the algorithm goes on with the next memeplex *Mem*$_{i+1}$ until all the memeplexes have been processed. Then, the memeplexes are merged to update the state of the population and the Pareto front (lines 20-21), proceeding with a new generation of the metaheuristic.

B. Integrating Parallelism

One of the most relevant features in MO-SFLA lies on the way the population is partitioned into memeplexes whose processing is carried out in an independent way at each generation of the metaheuristic. Its algorithmic design makes MO-SFLA a parallelism-aware metaheuristic which undertakes the solution of complex, time-consuming problems by conducting a parallel processing of the search space while also showing inherent parallel computing opportunities to accelerate execution time. More specifically, MO-SFLA shows two main levels of parallelism: 1) independent processing of different memeplexes (inter-memeplex parallelism), and 2) generation of new solutions within each memeplex (intra-memeplex parallelism). This two-level parallel design matches one of the current hardware trends: the two-level distributed+shared memory computing clusters, thus being suitable to be implemented by combining message-passing techniques with MPI (at the inter-memeplex level) and multithreading with OpenMP (at the intra-memeplex level). Figure 2 provides an overview of the parallel design of MO-SFLA, in which *m* MPI processes are spawned in accordance with the number of memeplexes following a master-worker scheme.

One of these processes assumes initially the role of master, carrying out the management of the data structures involved in the evolutionary process. At each generation, the master computes Pareto ranks and crowding distance values over the unsorted population P^u to sort it (via fast non-dominated sort and crowding ordering). The sorted population P^s is then partitioned into *m* memeplexes using the shuffling procedure, undertaking afterwards the assignment of memeplexes to the worker processes. Mem_i is assigned and copied into the memory of the ith worker by using `MPI_Send` (master side) and `MPI_Recv` (worker side). Once the assignment of memeplexes is complete, each worker conducts in parallel searches over its corresponding memeplex (in our implementation, the master assumes a worker role during this step to avoid idle resources). Upon termination of the worker tasks, their results are retrieved by the master, integrating them into the population and updating the Pareto front.

A first approach to parallelize the computations performed inside each memeplex is given in Algorithm 2. After the reception of the memeplex (lines 4-7 in Algorithm 2), the OpenMP work-sharing directive `#pragma omp for` is used to parallelize the processing of the memeplex using the available execution threads (lines 8-18). Each iteration of this parallel loop involves the different search strategies defined in MO-SFLA to generate new candidate solutions. Therefore, the time required by an iteration T_{it} can be modelled as follows:

$$T_{it}=\begin{cases} T_{lc} & \text{if}(P'_{lc} \succ Mem_{ij}), \\ T_{lc}+T_{gb} & \text{if}(!P'_{lc} \succ Mem_{ij} \&\& P'_{gb} \succ Mem_{ij}), \\ T_{lc}+T_{gb}+T_{ls} & \text{otherwise}. \end{cases} \tag{8}$$

In Equation 8, T_{lc}, T_{gb}, and T_{ls} refer to the time required by the generation and evaluation of a new solution from the best local individual, from the best global one, and from the local search, respectively, while P'_{lc}, P'_{gb} represent the candidates generated from the best local and best global individual. Therefore, Equation 8 denotes a variability in the execution time required by each iteration of the worker loop: while some iterations finish after generating P'_{lc}, others might stop after generating P'_{gb}, and others after the local search. This implies the presence of load imbalance in the parallel processing of a memeplex. This issue can have a noticeable impact in the speedup achievable at the parallel loop under static scheduling, since some threads may finish their assigned iterations (e.g., the ones which successfully generate an improved candidate solution from the best local individual in all their attempts) while others are still performing their computations. As the most time-consuming operations (objective functions) are involved in these computations, the resulting idle times can lead to a significant worsening in parallel performance.

Algorithm 2 Worker tasks in the initial parallel version of MO-SFLA

```
1:  MPI_Init /* initializing MPI Process #i */
2:  #pragma omp parallel num_threads (num_threads)
3:  while ! stop criterion is reached (maxEval) do
4:      #pragma omp single
5:          MPI_Recv ($Mem_i$, $n$, master_id)
6:          MPI_Recv ($Best_{global}$, 1, master_id)
7:          $Best_{local}$ ← Identify Best Local ($Mem_i$)
8:      #pragma omp for schedule (scheduleType)
9:      for $j$ = 1 to $n_l$ do
10:         $P'_{new}$ ← Learn from Best Local ($Best_{local}$, $Mem_{i(n-j)}$)
11:         if ! $P'_{new} \succ Mem_{i(n-j)}$ then
12:             $P'_{new}$ ← Learn from Best Global ($Best_{global}$, $Mem_{i(n-j)}$)
13:             if ! $P'_{new} \succ Mem_{i(n-j)}$ then
14:                 $P'_{new}$ ← Apply Local Search ($Mem_{i(n-j)}$)
15:             end if
16:         end if
17:         $Mem_{i(n-j)}$ ← $P'_{new}$
18:     end for
19:     #pragma omp single
20:         MPI_Send ($Mem_i$, $n$, master_id)
21: end while
```

Algorithm 3 Worker tasks in the counters-based parallel version of MO-SFLA

```
1:  MPI_Init /* initializing MPI Process #i */
2:  #pragma omp parallel num_threads (num_threads)
3:  while ! stop criterion is reached (maxEval) do
4:      #pragma omp single
5:          MPI_Recv ($Mem_i$, $n$, master_id)
6:          MPI_Recv ($Best_{global}$, 1, master_id)
7:          $Best_{local}$ ← Identify Best Local ($Mem_i$)
8:      #pragma omp for schedule (scheduleType)
9:      for $j$ = 1 to $n_l$ do
10:         switch ($Mem_{i(n-j)}$.counter)
11:             case 0: $P'_j$ ← Learn from Best Local ($Best_{local}$, $Mem_{i(n-j)}$)
12:             case 1: $P'_j$ ← Learn from Best Global ($Best_{global}$, $Mem_{i(n-j)}$)
13:             case 2: $P'_j$ ← Apply Local Search ($Mem_{i(n-j)}$)
14:         if $P'_j \succ Mem_{i(n-j)}$ || $Mem_{i(n-j)}$.counter == 2 then
15:             $Mem_{i(n-j)}$ ← $P'_j$, $Mem_{i(n-j)}$.counter ← 0
16:         else
17:             $Mem_{i(n-j)}$.counter ← $Mem_{i(n-j)}$.counter + 1
18:         end if
19:     end for
20:     #pragma omp single
21:         MPI_Send ($Mem_i$, $n$, master_id)
22: end while
```

Several strategies can be used to address the load imbalance problem. Initially, we can keep the original parallel scheme in Algorithm 2 and rely on the use of the OpenMP dynamic scheduling policies by setting the *scheduleType* in line 8 to 'dynamic'. Beyond that, the parallel design of MO-SFLA can be modified, e.g., to force each iteration of the memeplex processing loop to involve only one search regardless of the quality of the generated candidate solution. This idea is described in Algorithm 3, where a modified parallel scheme for MO-SFLA is proposed. The key idea behind this implementation lies on the use of counters to measure the number of trials an individual in the population has not been improved. The value in this counter decides the search strategy to be applied over the individual during the intra-memeplex processing (lines 10-13 in Algorithm 3). Given an individual Mem_{ij}, a value = 0 in $Mem_{ij}.counter$ will imply the generation of the new solution by taking as reference the best local individual in Mem_i, while a value = 1 will imply the learning from the best global individual, and a value = 2 a local search. In case of Mem_{ij}

being improved or local search, the counter is initialized to 0 and the solution stored in Mem_{ij} (lines 14-15). Otherwise, the new candidate is discarded and $Mem_{ij}.counter$ is increased (line 17) in such a way that, in a new generation of the metaheuristic (after merging and shuffling at the master), the worker will make a new attempt over Mem_{ij} with a different search strategy. Using this approach, T_{it} here is defined as:

$$T_{it}=\begin{cases} T_{lc} & \text{if}(Mem_{ij}.\text{counter} == 0), \\ T_{gb} & \text{if}(Mem_{ij}.\text{counter} == 1), \\ T_{ls} & \text{if}(Mem_{ij}.\text{counter} == 2). \end{cases} \quad (9)$$

V. Experimental Results and Evaluation

This section is focused on the experimental assessment of MO-SFLA. We will firstly detail experimental conditions and the metrics used to evaluate the proposal. Afterwards, we will conduct comparisons among the proposed parallel schemes and discuss the effect of each design over the observed speedups. In addition, we will analyze the results obtained by using different configurations of the algorithm in order to identify the most satisfying one attending to both parallel and multiobjective performance. Finally, the quality of the generated Pareto solutions will be assessed by making comparisons with other multiobjective and biological methods.

We have conducted experimentation by considering five real-world amino acid datasets[2], representing different problem sizes in terms of number of sequences and sequence length: 1) M67x11333: 67 sequences (11333 amino acids per sequence) of bacterial ancestry euBac proteins [43]; 2) M88x3329: 88 sequences (3329 amino acids per sequence) of MCM7 and RPB1/RPB2 from Thermophilic fungi [44]; 3) M187x814: 187 sequences (814 amino acids per sequence) of ABC-B transporters from Mycorrhiza-forming fungi [45]; 4) M260x1781: 260 sequences (1781 amino acids per sequence) of proto-oncogene MYB from Beta vulgaris [46]; and 5) M355x1263: 355 sequences (1263 amino acids per sequence) of DHA2, ARN, and GEX from hemiascomycete yeasts [47].

Our experiments were run on a hybrid distributed+shared-memory setup composed of four computing nodes interconnected by a Gigabit Ethernet network. Each node contains two AMD Opteron 6174 'Magny-Cours' twelve-core processors at 2.2 GHz with 12MB of L3 cache and 32GB of DDR3 RAM, running Ubuntu 14.04 LTS. The tested software was compiled by using GCC 5.2.1 with the -O3 optimization flag.

A. Performance Metrics and Statistical Methodology

Different performance metrics have been used to evaluate the results achieved by MO-SFLA. In order to measure parallel performance, we are using two well-known parallelism metrics: speedup and efficiency [48]. Let T_1 be the execution time on a single processing unit reported by an application and T_p the execution time on p processing units. The speedup SU measures the effective reduction in time achieved by the parallel version of the application ($\frac{T_1}{T_p}$), while the efficiency Eff calculates the average utilization of processing units ($\frac{SU(p)}{p}$). The calculation of these parallel metrics has been conducted by taking as T_1 the execution time reported, for each dataset, by the serial version of MO-SFLA (Table I).

[2]We used ProtTest [3] to choose the most fitting probabilistic model of sequence evolution to be considered in each dataset. According to its outputs, we used the LG+Γ model [41] on M67x11333, M88x3329, M187x814, and M355x1263, while M260x1781 was analyzed under the JTT+Γ model [42].

TABLE I
Serial execution time (in seconds) for each dataset

	M67x11333	M88x3329	M187x814
T_1	195059.48	57111.63	44171.92
	M260x1781	M355x1263	
T_1	66748.33	96219.62	

TABLE II
Normalization points for hypervolume calculations

	Ideal		Nadir	
Dataset	$P(T)$	$L(T)$	$P(T)$	$L(T)$
M67x11333	171196	-472077.06	199523	-497911.23
M88x3329	33406	-148722.11	34122	-151460.57
M187x814	29772	-133537.22	30620	-136763.70
M260x1781	43420	-163485.56	45119	-167893.13
M355x1263	54685	-230833.34	56040	-237019.23

For multiobjective performance, two metrics have been adopted [49]: hypervolume and set coverage. Given a Pareto approximation set X, the hypervolume I_H measures the k-dimensional volume of the objective space $\mathbb{R}^k$ (with regard to a point Z_{ref}) which is weakly-dominated by at least one $s \in X$, that is, the volume of the orthogonal polytope $\prod^k$:

$$\prod^k = \{p \in \mathbb{R}^k : p \preceq s \text{ for some } s \in X\}. \quad (10)$$

To avoid the influence of different objective scales, we normalize the scores of the solutions in X in the scale [0,1] by using the ideal and nadir points from Table II. These points correspond to the best (ideal) and the worst (nadir) objective scores observed (adding or subtracting a 0.25% of their values to allow room for future comparisons). After normalization, hypervolume values are calculated with regard to $Z_{ref} = (1,1)$, taking both objectives to be minimized by considering positive likelihood values. Regarding the second multiobjective metric, the set coverage SC compares two Pareto approximation sets X and Y by calculating the fraction of solutions in Y which are weakly-dominated by X:

$$SC(X,Y) = \frac{|\{y \in Y, \exists x \in X : x \succeq y\}|}{|Y|}. \quad (11)$$

Since we are studying stochastic methods, the obtained results samples have been examined under the following statistical tests (with a confidence level of 95%) to provide statistical reliability to the comparisons [50]. In a first step, the Kolmogorov-Smirnov normality test was used in order to check if the evaluated samples followed a Gaussian distribution. If so, we performed the analysis of homoscedasticity by using the Levene test, applying accordingly the ANOVA test in case of detecting homogeneity in variances. In case of dealing with non-Gaussian distributions or no homogeneity in variances, the analysis of statistically significant differences was conducted by using the Wilcoxon-Mann-Whitney test.

The configuration of input parameters in MO-SFLA was undertaken in the following way. Since the population size

TABLE III
EVALUATION OF PARALLEL PERFORMANCE (SPEEDUPS AND EFFICIENCIES) FOR EACH PARALLEL IMPLEMENTATION OF MO-SFLA

	16 cores		32 cores		64 cores	
	M67x11333					
Design	*SU*	*Eff* (%)	*SU*	*Eff* (%)	*SU*	*Eff* (%)
Static	10.39	64.94	16.92	52.88	26.80	41.88
Dynamic	11.66	72.88	20.13	62.91	30.02	46.91
Counters	**14.42**	**90.13**	**26.25**	**82.03**	**45.91**	**71.73**
	M88x3329					
Static	11.58	72.38	19.10	59.69	31.50	49.22
Dynamic	12.67	79.19	21.34	66.69	32.24	50.38
Counters	**14.94**	**93.38**	**28.18**	**88.06**	**53.76**	**84.00**
	M187x814					
Static	10.70	66.88	18.58	58.06	30.41	47.52
Dynamic	11.35	70.94	20.41	63.78	33.22	51.91
Counters	**14.75**	**92.19**	**27.71**	**86.59**	**53.54**	**83.66**
	M260x1781					
Static	12.75	79.69	20.45	63.91	31.57	49.33
Dynamic	13.45	84.06	21.97	68.66	32.40	50.63
Counters	**15.65**	**97.81**	**29.20**	**91.25**	**54.17**	**84.64**
	M355x1263					
Static	11.92	74.50	21.96	68.63	38.49	60.14
Dynamic	12.86	80.38	23.03	71.97	42.64	66.63
Counters	**15.68**	**98.00**	**29.87**	**93.34**	**54.24**	**84.75**

popSize shows influence over the remaining parameters, we firstly examined the hypervolume of the outputs generated when using a range of uniformly distributed values for *popSize*. According to the reported results, we set *popSize* to 128. The configuration of the remaining parameters (the number of memeplexes *m* and the number of learning steps per memeplex n_l) requires a more thorough analysis, since these parameters have an impact not only in multiobjective quality but also in parallel performance. This issue is addressed in the following subsections. As stop criterion, we considered a maximum number of 10000 evaluations in our experiments.

B. Comparison of Parallel Designs

In order to conduct the evaluation of MO-SFLA according to its best parallel implementation, we introduce firstly a comparison of parallel performance between the static version of the metaheuristic and the dynamic and counters-based implementations described in Section IV.B. For this purpose, we have measured the speedup and efficiency reported by each implementation considering growing system sizes (16, 32, and 64 cores), setting initially the values of *m* and n_l to 4 and 32 with the aim of fitting in the organization of our cluster setup. Table III shows the median speedups and efficiencies observed from 11 independent runs per experiment.

According to these results, we can verify that the static implementation attains satisfying speedups when considering low system sizes (in the range 10.4 - 12.8 for 16 cores). However, when moving to 32 cores the efficiency barely reaches the threshold of 60%. This issue is partially solved by the dynamic version, which leads to efficiencies up to 72.0% for 32 cores. In spite of this, both static and dynamic versions failed to achieve an accurate exploitation of parallel resources when considering 64 cores, showing efficiencies below 47% in the most time-consuming dataset (M67x11333). On the other hand, the counters-based approach gives rise to significant parallel performance in all the scenarios under evaluation. The attained efficiencies of 90.1% - 98.0% (16 cores), 82.0% - 93.3% (32 cores), and 71.7% - 84.8% (64 cores) suggest that this strategy is able to satisfactorily address the issues which affect the other parallel designs under study.

The time analysis of the median-speedup executions provides insight into the performance penalty introduced by the different sources of overhead in MO-SFLA: T_{critic} (non-parallelizable time, including serial sections of the algorithm and the time spent on MPI/OpenMP initializations and terminations), T_{comm} (communication time due to message passing at the memeplex assignment and results delivery sections), T_{sync}*(intra)* (synchronization times among OpenMP threads at the intra-memeplex level, including waiting times at the implicit barrier of `#pragma omp for`), and T_{sync}*(inter)* (synchronization times among MPI processes at the inter-memeplex level, referred to the times spent until a communication request is attended). The calculation of overhead time was conducted by using the OpenMP function *omp_get_wtime()*, which provides accurate time measurement in multi-threaded contexts. Table IV shows that the factor with the most impact in execution time is given by the two variations of T_{sync}. In the case of the dynamic implementation, the evolution of T_{sync}*(intra)* explains the worsening in efficiency shown when using high numbers of cores. In fact, the impact of T_{sync}*(intra)* over this implementation and the static one is very similar in the 64-core scenarios. This is due to the fact that the load balancing capabilities of the dynamic scheduling are related to the number of solutions to be generated by each execution thread inside their memeplex. By increasing the number of cores, the number of individuals (iterations) per thread is reduced, giving a narrower window to balance the memeplex processing. Under these circumstances, the dynamic approach does not fully solve the load imbalance issue in Equation 8.

On the contrary, the counters-based design leads to a sustainable reduction in the effect of T_{sync} both at the intra and inter-memeplex levels. In this approach, the time divergence during the memeplex processing is only governed by the differences between T_{lc}, T_{gl}, and T_{ls} (Equation 9) instead of depending on if-conditions that can introduce dramatic variations in T_{it} (Equation 8). It is possible then to reduce significantly the load imbalance inside each memeplex and, thereby, the inter-memeplex differences as the MPI processes can finish their tasks in a more equalized way. In fact, mean improvements of 78.8% and 72.9% are observed in T_{sync} with regard to the static and dynamic schemes through the use of the counters. Even though a rise in T_{critic} and T_{comm} is observed, these times do not have a significant effect over the overall execution time. Therefore, this analysis points out the counters-based implementation as an effective strategy to parallelize MO-SFLA, reporting accelerations up to 54x in the analysis of computationally demanding protein instances.

C. Multiobjective and Parallel Configuration

After determining the most satisfactory parallel implementation of MO-SFLA, we address the question of finding an optimal configuration of memeplexes (parameters *m* and n_l) for the metaheuristic attending to both multiobjective and parallel performance. For this purpose, we consider configurations involving *m*=2 and n_l=64 (named as 2x64), *m*=4 and n_l=32

TABLE IV
ANALYSIS OF OVERHEAD IMPACT (IN SECONDS) FOR EACH PARALLEL DESIGN OF MO-SFLA

	16 cores				32 cores				64 cores			
Design	T_{critic}	T_{comm}	$T_{sync}(intra)$	$T_{sync}(inter)$	T_{critic}	T_{comm}	$T_{sync}(intra)$	$T_{sync}(inter)$	T_{critic}	T_{comm}	$T_{sync}(intra)$	$T_{sync}(inter)$
M67x11333												
Static	1.01	5.10	2069.39	4200.04	0.90	5.04	2402.07	2562.71	0.96	5.25	2178.45	1509.46
Dynamic	0.98	4.83	1132.74	3470.67	0.94	4.87	1643.81	1681.14	0.93	4.88	1948.79	1073.56
Counters	1.04	6.01	**786.34**	**806.53**	1.24	6.63	**774.71**	**499.09**	1.26	6.12	**733.61**	**350.42**
M88x3329												
Static	1.14	4.75	646.69	423.93	1.07	4.49	659.58	300.80	1.23	5.23	652.43	173.83
Dynamic	0.98	4.87	400.61	295.80	1.09	4.65	537.70	172.99	1.23	4.95	640.65	126.17
Counters	1.23	5.82	**144.48**	**107.58**	1.24	6.08	**108.50**	**61.77**	1.25	6.07	**77.35**	**39.53**
M187x814												
Static	1.80	20.39	464.14	863.89	1.79	20.92	483.61	445.40	1.84	20.34	465.38	206.06
Dynamic	1.75	19.46	276.92	649.13	1.61	18.90	345.61	295.31	1.59	19.68	411.11	145.94
Counters	2.68	28.24	**101.74**	**74.33**	2.19	27.61	**86.97**	**45.55**	2.03	27.91	**65.44**	**23.73**
M260x1781												
Static	3.55	42.17	535.91	484.03	2.93	40.43	683.94	374.89	2.89	41.71	587.56	328.41
Dynamic	3.26	42.28	420.82	426.56	3.17	42.22	555.73	298.99	2.71	41.25	577.75	246.49
Counters	3.72	47.20	**131.31**	**119.13**	3.26	47.70	**128.00**	**75.40**	3.81	47.67	**106.53**	**56.52**
M355x1263												
Static	3.71	43.70	958.12	1213.95	3.34	45.10	829.02	712.46	3.63	45.26	794.27	216.41
Dynamic	3.74	43.08	404.77	1014.84	3.20	42.78	488.50	653.87	3.52	41.87	497.76	177.77
Counters	5.33	82.95	**270.09**	**224.81**	5.22	83.08	**265.01**	**115.55**	5.20	83.12	**255.21**	**59.20**

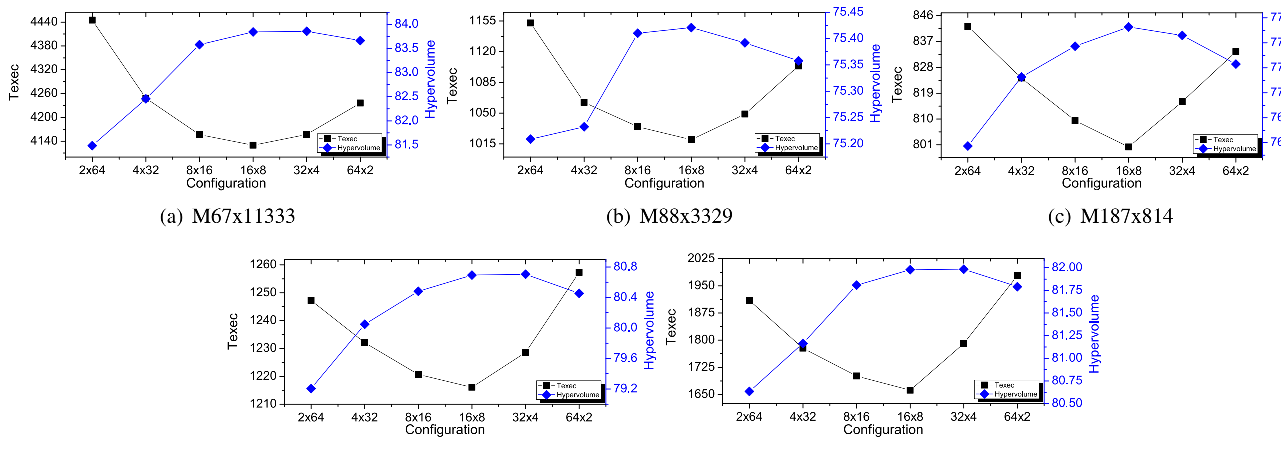


Fig. 3. Comparison of hypervolume scores and execution time for different configurations of m x n_l

TABLE V
PARALLEL AND MULTIOBJECTIVE RESULTS FOR DIFFERENT CONFIGURATIONS OF $m \times n_l$

	Multiobjective Metrics									
	M67x11333		M88x3329		M187x814		M260x1781		M355x1263	
Configuration	I_H (%)	SC_{all} (%)	I_H (%)	SC_{all} (%)	I_H (%)	SC_{all} (%)	I_H (%)	SC_{all} (%)	I_H (%)	SC_{all} (%)
2x64	81.49±0.20	17.45	75.21±0.12	13.33	76.47±0.20	14.58	79.20±0.18	10.43	80.63±0.20	15.18
4x32	82.46±0.59	19.28	75.23±0.04	21.64	77.16±0.35	30.91	80.05±0.19	22.36	81.16±0.20	15.84
8x16	83.58±0.58	51.86	75.41±0.04	72.03	77.47±0.46	32.49	80.48±0.14	41.15	81.81±0.11	51.54
16x8	83.84±0.44	58.75	**75.42±0.07**	**77.31**	**77.66±0.38**	**69.00**	80.69±0.23	71.94	81.98±0.18	72.50
32x4	**83.85±0.33**	**60.45**	75.39±0.04	75.00	77.58±0.58	67.51	**80.71±0.23**	**75.38**	**81.99±0.19**	**75.67**
64x2	83.66±0.33	51.99	75.36±0.07	46.79	77.29±0.58	41.95	80.45±0.11	45.67	81.79±0.21	56.79
	Parallel Metrics									
	SU	*Eff* (%)	*SU*	*Eff* (%)	*SU*	*Eff* (%)	*SU*	*Eff* (%)	*SU*	*Eff* (%)
2x64	43.64±0.89	68.19	49.54±1.33	77.41	52.45±0.33	81.95	53.50±0.40	83.59	50.17±0.81	78.39
4x32	45.91±1.17	71.73	53.76±1.39	84.00	53.54±0.45	83.66	54.17±0.38	84.64	54.24±1.14	84.75
8x16	46.93±1.55	73.33	55.20±1.49	86.25	54.57±0.53	85.27	54.68±0.24	85.44	56.69±0.78	88.58
16x8	**47.23±0.95**	**73.80**	**56.00±1.19**	**87.50**	**55.20±0.39**	**86.25**	**54.89±0.35**	**85.77**	**58.04±1.25**	**90.69**
32x4	46.93±1.17	73.33	54.44±1.49	85.06	54.13±0.35	84.58	54.33±0.23	84.89	53.85±1.30	84.14
64x2	46.05±1.26	71.95	51.75±1.56	80.86	53.00±0.28	82.81	53.09±0.35	82.95	48.75±1.40	76.17

(4x32), m=8 and n_l=16 (8x16), m=16 and n_l=8 (16x8), m=32 and n_l=4 (32x4), and m=64 and n_l=2 (64x2). Considering the results from 11 independent runs per configuration and dataset, we report in the upper side of Table V median hypervolume scores, as well as the results of calculating the set coverage of each configuration over the remaining ones in their median-hypervolume executions. The bottom side of Table V shows the median speedup values and the corresponding efficiencies attained by each configuration. We provide also in Figure 3 a graphical representation of the behaviour of each configuration in terms of hypervolume and execution time.

The analysis of multiobjective results points out the significant solution quality reported by the configurations 16x8 and 32x4, which obtain relevant hypervolume scores (in the interval 75.4% - 83.9%) in all the protein-based datasets under analysis. Additionally, the set coverage gives account of how these two configurations are able to dominate percentages up to 77% of the solutions generated by the remaining ones. Although our results give preference to the use of configurations with $m \geq 8$, the fact of increasing the number of memeplexes repeatedly does not guarantee the attainment of better results (as shown by the worsening introduced in the configuration 64x2). Particularly, a very high increase in the number of memeplexes will lead to a reduction in the number of individuals per memeplex, thus limiting the interactions at the intra-memeplex level defined by n_l. Therefore, from these results it can be observed the need to involve in the execution of the metaheuristic an accurate number of memeplexes to boost the parallel processing of the search space, considering a proper number of individuals inside each memeplex.

Although the multiobjective metrics point out the relevance of the configurations 16x8 and 32x4, the characteristics of MO-SFLA's design make mandatory the consideration of parallel performance in order to decide which configuration provides a better compromise between multiobjective quality and execution time. The evolution of execution times in Figure 3 and the parallel results reported in Table V show that the configurations 8x16 and 16x8 are the ones which best take advantage of parallelism. This is due to the fact that these configurations allow a better balance between the overhead introduced by OpenMP (which shows more influence over the configurations 2x64 and 4x32) and the overhead introduced by MPI (which has more impact over 32x4 and 64x2).

Therefore, the two sides of this analysis agree on highlighting the satisfactory behaviour achieved by the configuration 16x8 of the metaheuristic. This configuration is not only able to achieve relevant performance from a parallel perspective, but also from a multiobjective one. In fact, the statistical assessment of hypervolume samples (from 31 independent runs) for the configurations 16x8 and 32x4 reveals that the improvement observed when using 32x4 in M67x11333, M260x1781, and M355x1263 is not statistically significant.

D. Results Assessment

We undertake next the assessment of solution quality in MO-SFLA by introducing comparisons with other multiobjective and biological methods. For this purpose, we consider the results generated from 31 independent runs per experiment.

The first issue to be addressed is the comparison between the parallel and serial versions of the application. Since the parallel counters-based implementation introduces changes into the original algorithmic design of MO-SFLA, we need to check for a possible degradation of the solution quality. Table VI sheds light on this question, reporting the median hypervolume scores achieved in each dataset. The output of the statistical testing of hypervolume samples reports that no statistically significant differences were found between both versions, despite the slight improvement observed in the median scores of the parallel version. Hence, we can state that the counters-based approach allows MO-SFLA to maintain the solution quality of its original serial design in noticeably reduced time (from 54 hours to 69 minutes in the case of M67x11333).

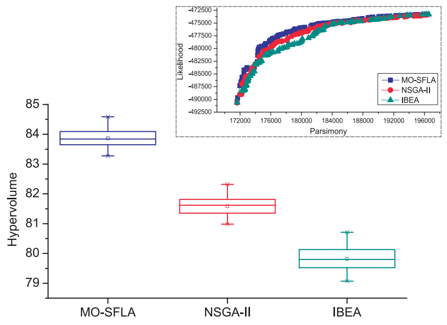
We now introduce comparisons of multiobjective quality with other multiobjective evolutionary algorithms: the widely-used NSGA-II and IBEA [51], [52], the latter being a metaheuristic that has previously shown significant performance in DNA contexts [53]. Both NSGA-II and IBEA[3] were parallelized using a hybrid MPI+OpenMP implementation based on the master-worker scheme, in which the most time-consuming computations identified in the time profiling of these algorithms were distributed among the worker processes and accelerated by using `#pragma omp for`. More specifically, the master carries out the management of populations, the identification of multiobjective quality (fast non-dominated sort and crowding in NSGA-II, indicator-based fitness assignment and environmental selection in IBEA), and the application of evolutionary operators to generate offspring distance matrices. Then, these matrices are distributed via MPI among the workers, who conduct the inference, topological optimization, and evaluation of the corresponding phylogenetic trees with OpenMP. These results are then transferred to the master, updating the Pareto front structure and beginning a new generation. Since we are using parallel versions of these metaheuristics, we also checked if the results reported by these implementations differed from the serial ones. According to the statistical analysis of hypervolume samples for NSGA-II and IBEA, non-significant differences were verified with regard to their original serial implementations.

TABLE VI
HYPERVOLUME I_H (%) COMPARISONS WITH THE SERIAL VERSION OF MO-SFLA (× = STAT. NON-SIGNIFICANT DIFFERENCES, ✓ = SIGNIFICANT DIFFERENCES)

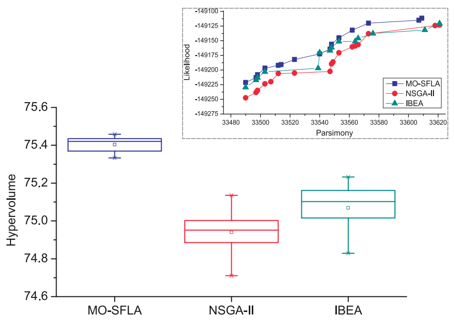
Dataset	Serial	Parallel	Diff.?
M67x11333	83.83±0.29	83.84±0.44	×
M88x3329	75.38±0.06	75.42±0.07	×
M187x814	77.40±0.49	77.66±0.38	×
M260x1781	80.62±0.20	80.69±0.23	×
M355x1263	81.92±0.13	81.98±0.18	×

Table VII provides median hypervolume values, set coverage scores and execution time for each metaheuristic (using 64 cores). A comparison of hypervolume box plots and Pareto fronts is given in Figure 4. Both hypervolume and set coverage highlight the quality of the solutions from MO-SFLA in comparison to NSGA-II and IBEA. On the one hand, the higher hypervolume values suggest that the Pareto fronts from MO-SFLA cover a wider area of the objective space, being the improvement over NSGA-II and IBEA statistically significant in all the datasets. By examining the Pareto fronts in Figure 4, we observe how MO-SFLA achieves better multiobjective performance (especially attending to the convergence property), resulting in mean set coverages over 90% (NSGA-II) and 79%

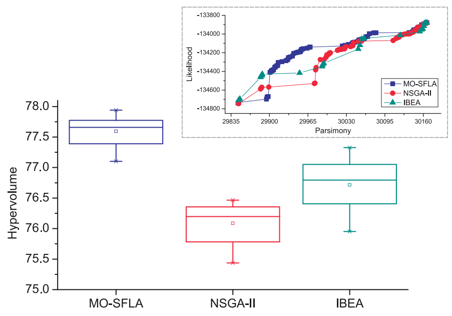
[3]The implementations of NSGA-II and IBEA include the same individual representation as in MO-SFLA with binary tournament selection, uniform crossover interchanging complete rows of the parent matrices [39], gamma-distributed mutation of matrix entries [53], and NNI/SPR/gradient optimization [40]. Fitness computations in IBEA use the hypervolume-based I_{HD} indicator [51]. In order to configure these algorithms, we examined the outputs generated from different configurations using hypervolume, checking for each input parameter different values uniformly distributed in the parameter range. For NSGA-II, the best parameter values were found to be: population size = 96, crossover probability = 70%, and mutation probability = 5%. For IBEA, we used a population size = 96, crossover probability = 70%, mutation probability = 5%, fitness scaling factor = 0.05, and I_{HD} reference point = (2,2). 10000 evaluations were set as stop criterion in both NSGA-II and IBEA.



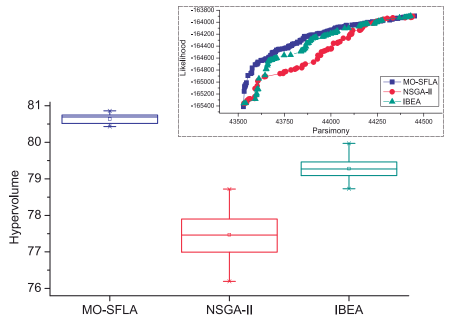
(a) M67x11333



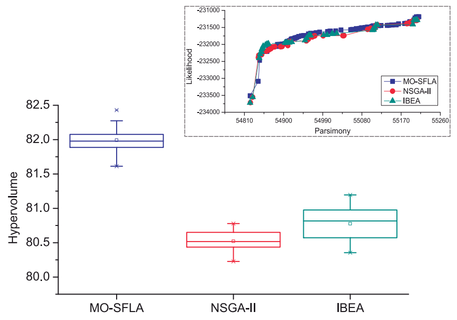
(b) M88x3329



(c) M187x814

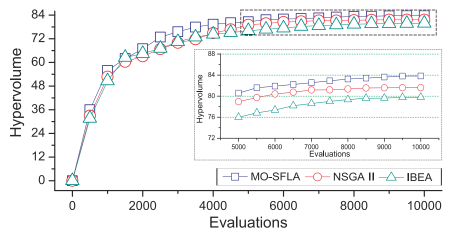


(d) M260x1781

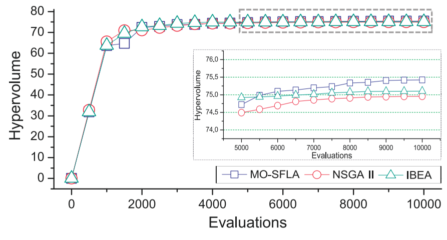


(e) M355x1263

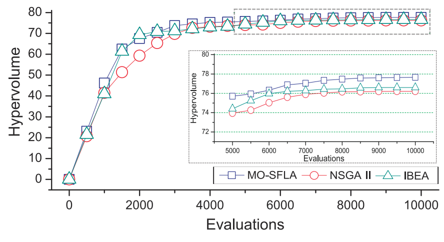
Fig. 4. Hypervolume box plots and Pareto fronts representation (from the median-hypervolume executions) for MO-SFLA, NSGA-II, and IBEA



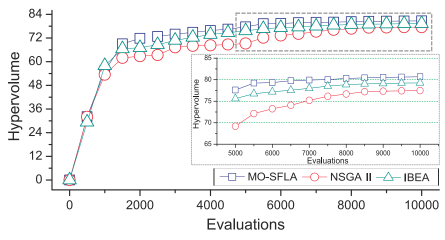
(a) M67x11333



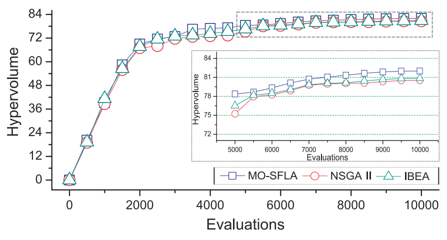
(b) M88x3329



(c) M187x814



(d) M260x1781



(e) M355x1263

Fig. 5. Convergence analysis of MO-SFLA, NSGA-II, and IBEA for the median-hypervolume executions

TABLE VII
MULTIOBJECTIVE COMPARISONS WITH NSGA-II AND IBEA (× = STAT. NON-SIGNIFICANT DIFFERENCES. ✓ = SIGNIFICANT DIFFERENCES)

	M67x11333		M88x3329		M187x814		M260x1781		M355x1263	
Metric	Score		Score		Score		Score		Score	
I_H(MO-SFLA) (%)	**83.84±0.44**		**75.42±0.07**		**77.66±0.38**		**80.69±0.23**		**81.98±0.18**	
I_H(NSGA-II) (%)	81.62±0.46	✓	74.95±0.12	✓	76.20±0.58	✓	77.46±0.91	✓	80.52±0.22	✓
I_H(IBEA) (%)	79.80±0.61	✓	75.10±0.15	✓	76.71±0.64	✓	79.27±0.39	✓	80.82±0.41	✓
SC(MO-SFLA,NSGA-II) (%)	**96.23**		**100.00**		**90.16**		**88.61**		**80.77**	
SC(NSGA-II,MO-SFLA) (%)	0.00		0.00		8.82		8.13		11.54	
SC(MO-SFLA,IBEA) (%)	**95.54**		**92.86**		**54.55**		**93.41**		**61.54**	
SC(IBEA,MO-SFLA) (%)	1.00		6.67		10.29		2.44		23.08	
T_{exec}(MO-SFLA) (s)	**4129.91**		**1019.79**		**800.26**		**1216.09**		**1661.60**	
T_{exec}(NSGA-II) (s)	5022.84		1259.14		1011.90		1475.17		2004.24	
T_{exec}(IBEA) (s)	5271.91		1328.40		1061.20		1498.41		2091.34	

TABLE VIII
BIOLOGICAL COMPARISONS WITH OTHER PHYLOGENETIC METHODS WITH SUPPORT FOR PROTEIN SEQUENCE DATA

	M67x11333		M88x3329		M187x814		M260x1781		M355x1263	
Method	$P(T)$	T-value	$P(T)$	T-value	$P(T)$	T-value	$P(T)$	T-value	$P(T)$	T-value
MO-SFLA	**171623**	**Best**	**33490**	**Best**	**29847**	**Best**	**43529**	**Best**	**54823**	**Best**
TNT	**171623**	**0.00**	**33490**	**0.00**	**29847**	**0.00**	**43529**	**0.00**	**54823**	**0.00**
ProtPars	173765	15.07	33944	9.28	29955	3.01	44495	14.88	55328	9.27
Method	$L(T)$	AU value	$L(T)$	AU value	$L(T)$	AU value	$L(T)$	AU value	$L(T)$	AU value
MO-SFLA	**-473260.21**	0.52	**-149094.84**	**0.71**	**-133871.90**	**0.57**	**-163895.81**	**0.65**	**-231186.40**	**0.74**
RAxML	**-473260.21**	0.49	**-149094.84**	0.64	-133879.69	0.37	-163918.74	0.26	-231201.40	0.43
IQ-TREE	**-473260.21**	**0.73**	**-149094.84**	0.32	-133871.96	0.55	-163899.10	0.57	-231301.11	0.12
GARLI	-473264.65	0.25	-149111.00	0.23	-133876.72	0.43	-163982.64	0.03	-231859.24	0.00
MrBayes	-474968.29	0.00	-149876.82	0.00	-134184.98	0.00	-166016.04	0.00	-233225.50	0.00

(IBEA). These results show the robustness of a metaheuristic engine involving multiple parallel searches via memeplexes, swarm-inspired strategies to generate solutions, and shuffling techniques to spread information among memeplexes.

The convergence analysis shown in Figure 5 sheds light on how multiobjective quality evolves throughout the execution of these three algorithms. It can be observed that MO-SFLA, NSGA-II, and IBEA are able to converge with the defined stop criterion in all the datasets under analysis. For the case of M67x11333, M187x814, M260x1781, and M355x1263, the interval of 8000-10000 evaluations represents the point where stable hypervolume scores are attained. In M88x3329, although convergence is achieved in earlier stages, improvements in the objective scores of the solutions in the front are still observed for the remaining evaluations.

Focusing on phylogenetic quality, Table VIII reports comparisons with biological tools from the literature. We have considered two maximum parsimony methods: the heuristic tool TNT [24] and ProtPars from the PHYLIP package [54]. The likelihood assessment has been conducted by using four methods with support for protein data: the reference tool for high-performance likelihood analyses RAxML [23], the stochastic hill-climbing method IQ-TREE [25], the genetic algorithm GARLI [20], and the Bayesian method MrBayes [26]. Despite also being an evolutionary algorithm, MetaPIGA [21] has not been included in the comparison since this tool does not currently include support for the LG+Γ model.

Due to the single-objective nature of these tools, we compare the extreme points of the median-hypervolume Pareto fronts from MO-SFLA with the median results obtained by each method from 31 independent runs per dataset. We used the parallel versions of these methods (when available) with input parameter configurations that matched the execution time of MO-SFLA. The comparison has been carried out by using two biological testing procedures [3]: the Kishino-Hasegawa (KH) test for parsimony and the CONSEL approximately unbiased (AU) test for likelihood. The KH test verifies if there are significant differences between solutions by calculating a T-value that represents the difference in the minimum number of substitutions on the phylogenies under comparison. CONSEL classifies the examined solutions by assigning a higher AU value to the phylogeny with higher statistical chance of representing the best likelihood hypothesis for the input data.

The upper side of Table VIII shows the comparison of parsimony results. We can observe that MO-SFLA is able to reach the parsimony quality of TNT in all the considered datasets, showing non-statistically significant differences attending to the output of the KH test (T-value = 0). As for ProtPars, KH reports a statistically significant worsening with regard to MO-SFLA and TNT. Regarding likelihood comparisons, the bottom side of Table VIII gives account of the significant performance of MO-SFLA, which obtains the highest AU value from CONSEL in four instances. In terms of likelihood scores, we can observe how the differences become more noticeable in datasets with a higher number of protein sequences (M260x1781 and M355x1263). Since the number of sequences defines the number of possible candidate solutions, such scores point out the improved processing of the phylogenetic search space attained by the algorithmic design of MO-SFLA. It is also worth mentioning that MO-SFLA is able to report in a single run an increased number of alternative maximum likelihood hypotheses in comparison to the remaining methods (e.g., 12 and 8 for M260x1781 and M355x1263, while the others infer 4 - RAxML - at most).

The attained solution quality gives account of the relevance of the search mechanisms in MO-SFLA, which tackles hard optimization scenarios according to three main strategies. Firstly, the algorithm deals with the problem of exploring multiple directions of huge search spaces by using the concept of memeplex to structure the population into partitions that evolve through the sharing of information between individuals. Secondly, at each memeplex, the generation of new candidate solutions takes into account the current status of the optimization process, accordingly using information from the best local or global individuals or from local search procedures. Thirdly, the incorporation of memeplex merging and shuffling strategies is aimed at enhancing the search capabilities of the algorithm, sharing the knowledge attained during the parallel searches through a uniform distribution of individuals among memeplexes at each generation of the algorithm.

In conclusion, this comparative evaluation suggests that MO-SFLA represents a step further in the solution of complex phylogenetic reconstructions in the protein domain. On the one hand, the additional computational complexity introduced by this kind of biological data has been addressed by means of an efficient parallel design which successfully exploits the computing capabilities of hybrid hardware setups. On the other hand, the definition of a search engine integrating different evolutionary strategies allows the proposal to conduct an effective processing of the search space, achieving significant solution quality on hard optimization scenarios involving increasing numbers of sequences in the input alignment.

VI. Conclusions

The adoption of more realistic assumptions in the modelling of optimization problems, along with the increased complexity observed in real-world data, have led to the need to design new approaches which effectively combine the advantages of bioinspired computing and parallelism. We have studied in this work the application of shuffled frog-leaping optimization to address the reconstruction of ancestral relationships in the protein domain. Our approach takes inspiration from the original SFLA design and integrates multiobjective optimization techniques to support a search engine based on the combination of swarm-based strategies, parallelism awareness in the shape of memeplexes, and shuffling techniques. Due to the time-consuming nature of the problem, we have studied the inclusion of parallelism by defining different parallel schemes aimed at the exploitation of multicore cluster platforms.

The proposed method, MO-SFLA, has been experimentally assessed over five real-world amino acid datasets by using a variety of parallel and multiobjective performance metrics. The comparison of the proposed parallel schemes has shown the advantages of associating trial counters to individuals to overcome load imbalance issues, minimizing idle time at the intra and inter-memeplex levels. In addition, we have undertaken the configuration of MO-SFLA taking into account both parallel and multiobjective perspectives, showing the impact of each configuration in solution quality and speedups to identify the one that provides the most satisfying overall behaviour. On the basis of these results, we have examined the quality of the solutions generated by MO-SFLA through comparisons with up to eight multiobjective and biological methods. While the hypervolume and set coverage highlight the improved convergence observed in the Pareto fronts, the biological testing points out the significant solution quality reported by the proposal, especially when complex datasets with high numbers of protein sequences are involved. The website http://arco.unex.es/sesaji/protein/ provides different resources related to the implementation and assessment of MO-SFLA.

Our future work lines are mainly aimed at exploiting additional parallelism opportunities in the proposal to accelerate the solution of very time-consuming datasets. More specifically, we will address the inclusion of new layers of parallelism e.g., at the objective function loop level, which can be efficiently accelerated by using GPUs and other co-processors on heterogeneous computing setups. In addition, we will propose alternative parallel designs of MO-SFLA for the exploitation of commodity platforms. Regarding its metaheuristic design, we will study the introduction of additional search strategies in the memeplexes with the aim of boosting solution quality.

Acknowledgment

This work was partially funded by the AEI (State Research Agency, Spain) and the ERDF (European Regional Development Fund, EU), under the contract TIN2016-76259-P (PROTEIN project), as well as Portuguese national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013. Sergio Santander-Jiménez is supported by the Post-Doctoral Fellowship from FCT under Grant SFRH/BPD/119220/2016.

References

[1] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 1–48, 2013.

[2] J. Pevsner, *Bioinformatics and Functional Genomics, 3rd Edition*. New Jersey: Wiley-Blackwell, 2015.

[3] P. Lemey, M. Salemi, and A.-M. Vandamme, *The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing*. Cambridge: Cambridge Univ. Press, 2009.

[4] A. Rokas, "Phylogenetic Analysis of Protein Sequence Data Using the Randomized Axelerated Maximum Likelihood (RAxML) Program," *Current Protocols in Molecular Biology*, vol. 96 (19.11), 2011.

[5] J. T. Cannon *et al.*, "Xenacoelomorpha is the sister group to Nephrozoa," *Nature*, vol. 530, pp. 89–93, 2016.

[6] M. M. Eusuff and K. E. Lansey, "Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm," *J. Water Res. Pl. ASCE*, vol. 129, no. 3, pp. 210–225, 2003.

[7] A. Sarkheyli, A. M. Zain, and S. Sharif, "The role of basic, modified and hybrid shuffled frog leaping algorithm on optimization problems: a review," *Soft Computing*, vol. 19, no. 7, pp. 2011–2038, 2015.

[8] T. G. Mattson, B. Massingill, and B. Sander, *Parallel Programming Patterns: Working with Concurrency in OpenMP, MPI, Java, and OpenCL*. Addison Wesley Professional, 2017.

[9] G. B. Fogel, "Evolutionary Computation for the Inference of Natural Evolutionary Histories," *IEEE Connect.*, vol. 3, no. 1, pp. 11–14, 2005.

[10] C. B. Congdon, "Phylogenetic Inference Using Evolutionary Algorithms," in *Computational Intelligence in Bioinformatics*. Wiley-IEEE Press, 2008, pp. 237–262.

[11] M. Sardaraz, M. Tahir, A. A. Ikram, and H. Bajwa, "Applications and Algorithms for Inference of Huge Phylogenetic Trees: a Review," *Am. J. Bioinformatics Res.*, vol. 2, no. 1, pp. 21–16, 2012.

[12] H. Matsuda, H. Yamashita, and Y. Kaneda, "Molecular Phylogenetic Analysis using both DNA and Amino Acid Sequence Data and Its Parallelization," *Genome Informatics*, vol. 5, pp. 120–129, 1994.

[13] H. Matsuda, "Protein phylogenetic inference using maximum likelihood with a genetic algorithm," in *Proc. of the Pacific Symposium on Biocomputing 96*. World Scientific, 1996, pp. 512–523.

[14] T. H. Reijmers, R. Wehrens, F. D. Daeyaert, P. J. Lewi, and L. M. C. Buydens, "Using genetic algorithms for the construction of phylogenetic trees: application to G-protein coupled receptor sequences," *Biosystems*, vol. 49, no. 1, pp. 31–43, 1999.

[15] K. Katoh, K. Kuma, and T. Miyata, "Genetic algorithm-based maximum-likelihood analysis for molecular phylogeny," *Journal of Molecular Evolution*, vol. 53, no. 4-5, pp. 477–484, 2001.

[16] S. L. K. Pond and S. D. W. Frost, "A Genetic Algorithm Approach to Detecting Lineage-Specific Variation in Selection Pressure," *Mol. Biol. Evol.*, vol. 22, no. 3, pp. 478–485, 2005.

[17] T. Hill, A. Lundgren, R. Fredriksson, and H. B. Schiöth, "Genetic algorithm for large-scale maximum parsimony phylogenetic analysis of proteins," *Biochim. Biophys. Acta.*, vol. 1725, no. 1, pp. 19–29, 2005.

[18] J. J. Tapia and E. E. Vallejo, "A clustering genetic algorithm for inferring protein-protein functional interactions from phylogenetic profiles," in *Proc. of IEEE CEC 2008*. IEEE, 2008, pp. 2757–2763.

[19] D. J. Zwickl, *Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets Under the Maximum Likelihood Criterion. Ph.D. Thesis*. USA: Univ. Texas at Austin, 2006.

[20] A. L. Bazinet, D. J. Zwickl, and M. P. Cummings, "A Gateway for Phylogenetic Analysis Powered by Grid Computing Featuring GARLI 2.0," *Systematic Biology*, vol. 63, no. 5, pp. 812–818, 2014.

[21] R. Helaers and M. Milinkovitch, "MetaPIGA v2.0: maximum likelihood large phylogeny estimation using the metapopulation genetic algorithm and other stochastic heuristics," *BMC Bioinformatics*, vol. 11, no. 1, pp. 379–389, 2010.

[22] A. Stamatakis, "An Efficient Program for Phylogenetic Inference Using Simulated Annealing," in *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2005, pp. 1–8.

[23] A. Stamatakis, "RAxML Version 8: A Tool for Phylogenetic Analysis and Post-Analysis of Large Phylogenies," *Bioinformatics*, vol. 30, no. 9, pp. 1312–1313, 2014.

[24] P. A. Goloboff and S. A. Catalano, "TNT version 1.5, including a full implementation of phylogenetic morphometrics," *Cladistics*, vol. 32, no. 3, pp. 221–238, 2016.

[25] L. Nguyen, H. Schmidt, A. Haeseler, and B. Minh, "IQ-TREE: A fast and effective stochastic algorithm for estimating maximum likelihood phylogenies," *Mol. Biol. Evol.*, vol. 32, no. 1, pp. 268–274, 2015.

[26] F. Ronquist *et al.*, "MrBayes 3.2: Efficient Bayesian Phylogenetic Inference and Model Choice Across a Large Model Space," *Systematic Biology*, vol. 61, no. 3, pp. 539–542, 2012.
[27] L. Poladian and L. Jermiin, "Multi–Objective Evolutionary Algorithms and Phylogenetic Inference with Multiple Data Sets," *Soft Computing*, vol. 10, no. 4, pp. 359–368, 2006.
[28] S. Santander-Jiménez and M. A. Vega-Rodríguez, "Performance Evaluation of Dominance-Based and Indicator-Based Multiobjective Approaches for Phylogenetic Inference," *Information Sciences*, vol. 330, pp. 293–314, 2016.
[29] G. P. Coelho, A. E. A. Silva, and F. J. V. Zuben, "An Immune-Inspired Multi–Objective Approach to the Reconstruction of Phylogenetic Trees," *Neural Comput. Appl.*, vol. 19, no. 8, pp. 1103–1132, 2010.
[30] W. Cancino and A. C. B. Delbem, "A Multi–Criterion Evolutionary Approach Applied to Phylogenetic Reconstruction," in *New Achievements in Evol. Comp.* InTech, 2010, pp. 135–156.
[31] V. Jayaswal, L. Poladian, and L. S. Jermiin, "Single- and multi-objective phylogenetic analysis of primate evolution using a genetic algorithm," in *Proc. of IEEE CEC 2007*. IEEE, 2007, pp. 4146–4153.
[32] J. R. Macey, "Plethodontid salamander mitochondrial genomics: A parsimony evaluation of character conflict and implications for historical biogeography," *Cladistics*, vol. 21, no. 2, pp. 194–202, 2005.
[33] S. Santander-Jiménez and M. A. Vega-Rodríguez, "Applying a Multiobjective Metaheuristic Inspired by Honey Bees to Phylogenetic Inference," *BioSystems*, vol. 114, no. 1, pp. 39–55, 2013.
[34] W. H. E. Day, D. S. Johnson, and D. Sankoff, "The Computational Complexity of Inferring Rooted Phylogenies by Parsimony," *Mathematical Biosciences*, vol. 81, no. 1, pp. 33–42, 1986.
[35] B. Chor and T. Tuller, "Maximum likelihood of evolutionary trees: hardness and approximation," *Bioinformatics*, vol. 21, pp. 97–106, 2005.
[36] J. Lin, Y. Zhong, and J. Zhang, "A modified discrete shuffled flog leaping algorithm for RNA secondary structure prediction," in *Advances in control and communication*. Springer, 2012, pp. 591–599.
[37] B. Hu *et al.*, "Feature Selection for Optimized High-dimensional Biomedical Data using an Improved Shuffled Frog Leaping Algorithm," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, pp. 1–10, DOI: 10.1109/TCBB.2016.2602263, 2016.
[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multi–Objective Genetic Algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.
[39] L. Poladian, "A GA for maximum likelihood phylogenetic inference using neighbour-joining as a genotype to phenotype mapping," in *Genetic and Evolutionary Computation Conference*, 2005, pp. 415–422.
[40] A. Goëffon, J. M. Richer, and J. K. Hao, "Progressive Tree Neighborhood Applied to the Maximum Parsimony Problem," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 5, no. 1, pp. 136–145, 2008.
[41] S. Q. Lee and O. Gascuel, "An improved general amino acid replacement matrix," *Mol. Biol. Evol.*, vol. 25, no. 7, pp. 1307–1320, 2008.
[42] D. T. Jones, W. R. Taylor, and J. M. Thornton, "The rapid generation of mutation data matrices from protein sequences," *Bioinformatics*, vol. 8, no. 3, pp. 275–282, 1992.
[43] D. He, O. Fiz-Palacios, C. Fu, J. Fehling, C. Tsai, and S. L. Baldauf, "An Alternative Root for the Eukaryote Tree of Life," *Current Biology*, vol. 24, no. 4, pp. 465–470, 2014.
[44] I. Morgenstern *et al.*, "A molecular phylogeny of thermophilic fungi," *Fungal Biology*, vol. 116, no. 4, pp. 489–502, 2012.
[45] A. Kovalchuk, A. Kohler, F. Martin, and F. O. Asiegbu, "Diversity and evolution of ABC proteins in mycorrhiza-forming fungi," *BMC Evolutionary Biology*, vol. 15, no. 249, pp. 1–19, 2015.
[46] R. Stracke *et al.*, "Genome-wide identification and characterisation of R2R3-MYB genes in sugar beet (Beta vulgaris)," *BMC Plant Biology*, vol. 14, no. 249, pp. 1–17, 2014.
[47] P. J. Dias and I. Sá-Correia, "The drug:H+ antiporters of family 2 (DHA2), siderophore transporters (ARN) and glutathione:h+antiporters (GEX) have a common evolutionary origin in hemiascomycete yeasts," *BMC Genomics*, vol. 14, no. 901, pp. 1–22, 2013.
[48] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach, 5th Edition*. Morgan Kaufmann Publishers Inc., 2011.
[49] C. Coello, C. Dhaenens, and L. Jourdan, *Advances in Multi-Objective Nature Inspired Computing*. Berlin / Heidelberg: Springer Verlag, 2010.
[50] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures. 5th edition*. NY, USA: Chapman & Hall/CRC, 2011.
[51] E. Zitzler and D. Brockhoff, "Indicator-Based Evolutionary Algorithm," http://www.tik.ee.ethz.ch/sop/pisa/selectors/ibea/?page=ibea.php, 2014.
[52] E. Zitzler and S. Künzli, "Indicator-Based Selection in Multiobjective Search," in *Parallel Problem Solving From Nature VIII*, ser. LNCS, vol. 3242. Springer Verlag, 2004, pp. 832–842.
[53] S. Santander-Jiménez and M. A. Vega-Rodríguez, "Inferring Multiobjective Phylogenetic Hypotheses by Using a Parallel Indicator-Based Evolutionary Algorithm," in *Theory and Practice of Natural Computing*, ser. LNCS, vol. 8890. Springer Verlag, 2014, pp. 205–217.
[54] J. Felsenstein, "PHYLIP (phylogeny inference package)," http://evolution.genetics.washington.edu/phylip.html, 2000.

Sergio Santander-Jiménez received the Ph.D. degree in Computer Engineering from the University of Extremadura, Spain, in 2016. He is currently a postdoctoral fellow at the R&D Instituto de Engenharia de Sistemas e Computadores (INESC-ID), Instituto Superior Técnico (IST), Universidade de Lisboa (UL), Portugal. He has co-organized several international workshops on high-performance computing, computational intelligence, computational biology and bioinformatics, reviewing articles on these topics for different international JCR-indexed journals. His main research interests include evolutionary and bioinspired computing, multi-objective optimization, parallel and distributed computing, and their applications to real-world biological problems.

Miguel A. Vega-Rodríguez received the Ph.D. degree in Computer Engineering from the University of Extremadura, Spain, in 2003. He is currently an Associate Professor (accredited as Full Professor) of computer architecture in the Department of Computer and Communications Technologies, University of Extremadura. He has authored or co-authored more than 610 publications including journal papers (more than 110 JCR-indexed journal papers), book chapters, and peer-reviewed conference proceedings, for which he got several awards - such as Best Paper Awards in ISDA'11, IBERGRID'11, ICEC'09, and IEA-AIE'08. He has contributed to the organization of several international conferences and workshops, namely as general chair or co-chair. He has edited 10 special issues of international JCR-indexed journals. In addition, he is an editor and a reviewer of diverse international JCR-indexed journals. His main research interests include parallel and distributed computing, evolutionary computation, bioinformatics, and reconfigurable and embedded computing.

Leonel Sousa received the Ph.D. degree in Electrical and Computer Engineering from the Instituto Superior Técnico (IST), Universidade de Lisboa (UL), Lisbon, Portugal, in 1996. He is currently a Full Professor with UL and a Senior Researcher with the R&D Instituto de Engenharia de Sistemas e Computadores (INESC-ID). He has authored or coauthored more than 200 papers in journals and international conferences, and has edited four special issues of international journals. His research interests include VLSI architectures, computer architectures and arithmetic, parallel computing, and signal processing systems. Prof. Sousa is a Fellow of the IET and a Distinguished Scientist of the ACM. He has contributed to the organization of several international conferences as Program Chair and as General and Topic Chair. He is Associate Editor of the IEEE TMM, IEEE TCSVT, IEEE Access, IET Electronics Letters, Springer JRTIP, and Editor-in-Chief of the Eurasip JES. He was the recipient of several awards, including the DASIP'13 Best Paper Award, the SAMOS'11 'Stamatis Vassiliadis' Best Paper Award, the DASIP'10 Best Poster Award, and Honorable Mention Awards from the Universidade Técnica de Lisboa/Santander Totta (2007, 2009) and the Universidade de Lisboa/Santander (2016).