Parallel Multi-Island Genetic Algorithms for Sorting Unsigned Genomes by Reversals

Lucas A. da Silveira[†], José L. Soncco-Álvarez[*], Thaynara A. de Lima[‡] and Mauricio Ayala-Rincón[†]
[†]Department of Computer Science, Universidade de Brasília, 70900–010 Brasília D.F., Brazil
[*]Departamento Académico de Informática, Universidad Nacional de San Antonio Abad del Cusco, Cusco, Perú
[‡]Institute of Mathematics and Statistics, Universidade Federal de Goiás — Campus II, 74690–900 Goiânia, Brazil

***Abstract*—Sorting unsigned permutations by reversals is an $\mathcal{NP}$-hard optimization problem with applications in computational molecular biology. Several approximation and metaheuristic algorithms were proposed, among them, in a previous work, a competitive genetic algorithm and its parallel version using island models were proposed. In this paper, focusing on improving accuracy, new island models are proposed by diversifying the distribution of genetic material between islands through static and dynamic communication topologies. In static topologies, communication between islands is predefined and maintained during the computation, while in dynamic topologies the communication is continuously modified.**

The proposed island models use parallelism in a global and a local level, in which respectively occurs the exchange of individuals between islands and the fitness computation.

Results from the experiments performed with randomly generated synthetic permutations show that parallel island models using both dynamic and static communication topologies outperform parallel approaches found in the literature in terms of speed-up as well as accuracy.

I. Introduction

Computing the evolutionary distance between organisms through the comparison of their genomes has become possible as consequence of the development of computational molecular biology and modern genetics in the last decades. Such *evolution* among organisms proceed in two different manners: as local mutations and as global rearrangements. In the former case, nucleotide substitutions, deletions and insertions, result in local changes in the DNA sequence. In the latter case, segments (genes) of the chromosome are rearranged using some evolutionary operation. The most common genome rearrangement operations are: reversals, translocations, fusions, and block exchanges. In this work, we will consider reversals for calculating the evolutionary distance between organisms containing a single chromosome. In the literature, computing the minimum number of reversals for transforming one genome into another is known as the *reversal distance problem*. The complexity of this problem may vary depending on how the genes are abstracted into the genome. If the orientation of the genes is considered, the problem is known as the *signed reversal distance problem* (for short, *SRD*) and belongs to the class $\mathcal{P}$; otherwise, if the orientation of the genes is not considered, the problem is known as the *unsigned reversal distance problem* (for short, *URD*) and is known to be $\mathcal{NP}$-hard.

For the case of *SRD*, Hannenhalli and Pevzner [1] proposed a quadratic run-time algorithm to compute the reversal distance. Berman and Hannenhalli [2] improved this complexity obtaining an $O(n\alpha(n))$ algorithm, where n is the length of the input genome and $\alpha(n)$ is the inverse of Ackerman's function, and finally, Bader et al. [3] proposed a linear time algorithm.

Regarding the *URD* problem, Caprara [4] proved its $\mathcal{NP}$-hardness. Before Caprara's proof, Kececioglu and Sankoff [5] proposed an approximation algorithm of ratio 2.0 further improved by Bafna and Pevzner [6] to ratio 1.75. Subsequently, Christie [7] improved the ratio to 1.5, and finally, Berman et al. [8] proposed a solution with approximation ratio of 1.375. The last algorithm involves a complex theoretical background, and so far no implementation is known. The first Genetic Algorithm (GA) for solving *URD* was proposed by Auyeung and Abraham [9]; their method uses a population of size n^2 with time complexity in $O(n^5)$. Subsequently, a GA was proposed by Garrarizadeh et al. [10] which uses a population of size $n\log(n)$ with complexity of $O(n^4\log^2(n))$. After that, in [11], Soncco-Álvarez and Ayala-Rincón proposed a simple GA approach with complexity $O(n^3\log n)$ which was competitive and that with further adjusts became what here is called the standard (sequential) GA ($\mathcal{GA}_\text{s}$) [12], and provided better results than those computed by the approximation algorithm of ratio 1.5. Subsequently, parallel GAs came into play. In this new scenario, solutions deal with small populations running independently as proposed in [13], as well as performing the exchange of individuals among the populations involved at fixed intervals, as proposed in [14].

The main contributions of this work are new island models to solve the *URD* problem. These models explore static and dynamic communication topologies focusing on their influence in performance and accuracy. In [14], the authors proposed island models with static communication topologies for which an unidirectional ring topology provided the better known results. In this work, we are proposing new island models with static communication over topologies such as torus, 4×3-graph and tree. Also, dynamic communication topologies are proposed. In static topologies, communication between islands is predefined and maintained during the computation, while in dynamic topologies the communication is continuously modified. In this manner, the former models establish a rigid migration policy, while the latter ones a variable migration process which changes generation after generation. In the dynamic

models, each island is qualified according to fitness and diversity of its population, and then, at running time, interactions between islands are established before the migration phase occurs. In the search for a better performance, improvements on the proposed models were done through parallelism in two levels. In the first level, for the communication between islands, and in the second one, inside each island, for the parallel computation of the fitness function of individuals. From the experiments and statistical validation, it is possible to observe both performance and accuracy gains in solutions provided by the new parallel island models, especially those that use 4×3-graph topology and dynamic topology, when compared with the island model that uses the unidirectional ring topology proposed in [14].

Section II presents the necessary notions and properties involved in the reversal distance problem; Section III describes the evolutionary engine responsible for the breeding cycle in the proposed island models and presents the island models found in literature to solve *URD*; Section IV presents the proposed island models to resolve *URD*; Experiments and results are given in Section V; Section VI discusses the experiments and presents statistical tests to validate the results; Finally, Section VII concludes and discusses future work. For the benefit of the reviewing process, the source code of our proposed island models and data used for experiments are available at `genoma.cic.unb.br`.

II. Background

A. Definitions and Terminology

A genome can be represented as a special finite sequence of integers whose different elements are the different genes of the genome; namely, a uni-chromosomal genome of length $n \in \mathbb{N}$, is represented by $\pi = \pi(1) \dots \pi(n)$, where $|\pi(i)| \neq |\pi(j)|$, if $i \neq j$. As discussed in the Introduction, there are two situations of interest: *URD* and *SRD*, to be formalized below.

Situation 1 (*URD*): $\pi(i) \in [n] := \{1, \dots, n\}$, $1 \leq i \leq n$. In this case, notice that π can be seen as the bijective function

$$\pi : \quad [n] \quad \to \quad [n] \quad \text{such that} \quad i \quad \mapsto \quad \pi(i)$$

where $\pi(i)$ represents the element at i^{th} position of the sequence π; indeed, π is a permutation over the set $[n]$, which implies that π is an element of the well-known symmetric group S_n with composition as operator function. In S_n, the *inverse* of a permutation π is a permutation, denoted as π^{-1}, such that for all $i, j \in [n]$, $\pi(i) = j$ iff $\pi^{-1}(j) = i$. Denote by id_n the the *identity* permutation on S_n: $id_n(i) = i$, for all $i \in [n]$; thus, $\pi\pi^{-1} = \pi^{-1}\pi = id_n$. In the following, for simplicity, we will suppress the subscript n of id_n. A permutation π over $[n]$ is called an *unsigned permutation* of length n. The *reversal* $\rho_{(i,j)}$, $1 \leq i \leq j \leq n$, applied over an unsigned permutation π, reverses the elements from $\pi(i)$ to $\pi(j)$ into the permutation π, that is,

$$\pi\rho_{(i,j)} = \pi(1) \dots \pi(i-1) \boxed{\pi(j) \dots \pi(i)} \pi(j+1) \dots \pi(n)$$

Thus, reversals are also permutations in S_n; indeed, $\rho_{(i,j)}$ is the permutation such that $\rho_{(i,j)}(k) = k$, for $1 \leq k < i$ or $j < k \leq n$, and $\rho_{(i,j)}(k) = (j+i) - k$, for $i \leq k \leq j$. Given two permutations π and σ over $[n]$, the *reversal distance* between π and σ is defined as the minimum number of reversals needed to transform π into σ. Notice that d is the reversal distance between π and σ iff d is the reversal distance between $\sigma^{-1}\pi$ and id. In fact, if $\rho_1, \dots, \rho_d$ are reversals (even, permutations), we have that $\pi\rho_1 \cdots \rho_d = \sigma$ iff $\sigma^{-1}\pi\rho_1 \cdots \rho_d = \sigma^{-1}\sigma = id$.

From the previous observation, it makes sense to define the *URD* problem as follows: given an unsigned permutation π, find the reversal distance between π and id; such number is denoted by $d(\pi)$. For example, considering the 5-length permutation $\pi = 4\ 2\ 1\ 5\ 3$, $d(\pi) = 3$ and a minimal sequence of reversals that transform π into id is illustrated in Figure 1.

$$\begin{array}{rclcrcl} \pi & = & \underline{4\ 2\ 1}\ 5\ 3 & & \pi\rho_{(1,3)}\rho_{(3,5)} & = & 1\ 2\ 3\ \underline{5\ 4} \\ \pi\rho_{(1,3)} & = & 1\ 2\ \underline{4\ 5\ 3} & & \pi\rho_{(1,3)}\rho_{(3,5)}\rho_{(4,5)} & = & 1\ 2\ 3\ 4\ 5 \end{array}$$

Fig. 1. Sequence of reversals that sort the permutation $\pi = 4\ 2\ 1\ 5\ 3$

Since id is *sorted*, it will also be said that transforming π by reversals into id is "*sorting* π by reversals".

Situation 2 (*SRD*): $\pi(i) \in [\pm n] := \{\pm 1, \dots, \pm n\}$, $1 \leq i \leq n$. In this case, π is called a *signed permutation* of length n.

The action of a reversal $\rho_{(i,j)}$, $1 \leq i \leq j \leq n$, over a signed permutation π not only reverses the elements from $\pi(i)$ to $\pi(j)$, but also switches their signs:

$$\pi\rho_{(i,j)} = \pi(1) \dots \pi(i-1) \boxed{-\pi(j) \dots -\pi(i)} \pi(j+1) \dots \pi(n)$$

The reversal distance between signed permutations on $[\pm n]$ is defined as for *URD*, as the minimum number of reversals necessary to transform one permutation into the other. Denote by id_+ the signed permutation such that $id_+(i) = +i$, $1 \leq i \leq n$. Given a signed permutation π, the *Signed Reversal Distance Problem* consists in finding the reversal distance between π and id_+. For each signed permutation π over $[\pm n]$ one can associate an unsigned permutation $\pi' = \pi'(1) \dots \pi'(2n)$ over $[2n]$ as follows: if $\pi(i) \in \mathbb{Z}_+$ then $\pi'(2i-1) := 2\pi(i) - 1$ and $\pi'(2i) := 2\pi(i)$; otherwise, if $\pi(i) \in \mathbb{Z}_-$ then $\pi'(2i-1) := -2\pi(i)$ and $\pi'(2i) := -2\pi(i) - 1$. By this correspondence, the set of all signed permutations can be seen as a subgroup of the symmetric group S_{2n} (see [15]), where the allowable reversals on π' have the form $\rho_{(2i-1,2j)}$, $1 \leq i \leq j \leq n$. The example in Figure 2 illustrates this transformation and how reversals on the unsigned permutation are mimicked on its transformation.

$$\begin{array}{rcl} \pi = \underline{-2\ +3\ +4}\ -1 & & \pi' = \underline{4\ 3\ 5\ 6\ 7\ 8}\ 2\ 1 \\ \pi\rho_1 = \underline{-4\ -3\ +2\ -1} & & \pi'\rho'_1 = \underline{8\ 7\ 6\ 5\ 3\ 4\ 2\ 1} \\ \pi\rho_2 = +1\ \underline{-2}\ +3\ +4 & \rightsquigarrow & \pi'\rho'_2 = 1\ 2\ \underline{4\ 3}\ 5\ 6\ 7\ 8 \\ \pi\rho_3 = +1\ +2\ +3\ +4 & & \pi'\rho'_3 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \end{array}$$

Fig. 2. Sorting by reversals, $\pi = -2\ +3\ +4\ -1$ and its unsigned transformation π'. The reversals $\rho_1 = \rho_{(1,3)}$, $\rho_2 = \rho_{(1,4)}$ and $\rho_3 = \rho_{(2,2)}$ are mimicked by $\rho'_1 = \rho_{(1,6)}$, $\rho'_2 = \rho_{(1,8)}$ and $\rho'_3 = \rho_{(3,4)}$, respectively.

An important data structure to solve reversal distance problems is the *breakpoint graph*. Consider a permutation π over $[n]$. Add to π the pivots $\pi(0) = 0$ and $\pi(n+1) = n+1$.

For $0 \leq i \leq n$, if $|\pi(i) - \pi(i+1)| = 1$ then the unsorted pair $\{\pi(i), \pi(i+1)\}$ is called an *adjacency*, otherwise, $\{\pi(i), \pi(i+1)\}$ is called a *breakpoint*. The *breakpoint graph* $G(\pi)$ of π is a bi-colored graph such that: the set of vertices is $\{\pi(0), \pi(1), \ldots, \pi(n), \pi(n+1)\}$, there is a black edge between $\pi(i)$ and $\pi(i+1)$ iff $\{\pi(i), \pi(i+1)\}$ is a breakpoint and, a gray edge between $\pi(i)$ and $\pi(i)+1$ iff $\{i, \pi^{-1}(\pi(i)+1)\}$ is not an adjacency, for $0 \leq i \leq n$. Denote by $b(\pi)$ the number of breakpoints of π (or the number of black edges of $G(\pi)$) and $c(\pi)$ the maximum number of alternating cycles in $G(\pi)$, where alternating paths have edges with alternating (black-gray) colors (See example in Figure 3).

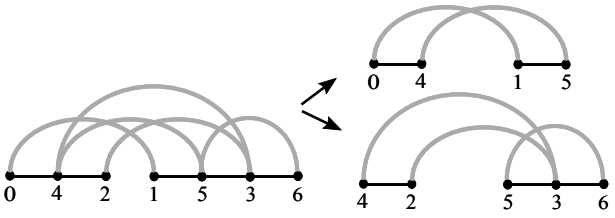

Fig. 3. Breakpoint graph for $\pi = 4\ 2\ 1\ 5\ 3$ (on the left) and a maximum decomposition into 2 alternating cycles (graphs on the right). Notice that $b(\pi) = 5$ and $c(\pi) = 2$

Bafna and Pevzner [6] showed that for unsigned permutations $b(\pi) - c(\pi) \leq d(\pi)$. Caprara [16] proved that the probability that $b(\pi) - c(\pi)$ is not equal to $d(\pi)$ is asymptotically $\Theta(\frac{1}{n^5})$ for a random unsigned permutation over $[n]$ and $\Theta(\frac{1}{n^2})$ for a random signed permutation over $[\pm n]$. Notice that for an unsigned permutation π that is image of a signed permutation, it is easy to calculate $c(\pi)$ because each vertex of $G(\pi)$ has degree at most two; however, for arbitrary unsigned permutations, determining such number is a difficult task. The previous are key observations to prove that *SRD* belongs to $\mathcal{P}$, [17], whereas *URD* is an $\mathcal{NP}$-hard problem [4]. Thus, algorithms that provide approximated solutions for *URD* are of great interest. Some papers in the literature present heuristics that use exact solutions of *SRD* in order to build approximated solutions for *URD* ([9], [12], [18], [19]). The parallel GAs proposed in the current work can be classified in this category. Indeed, our parallel GAs intensively apply the genetic algorithm $\mathcal{GA}_s$ given in [12]. Basically, the initial steps of $\mathcal{GA}_s$ ([12]) consist in given an input unsigned permutation π over $[n]$, associate to π, $n \log n$ randomly signed versions of π that are permutations over $[\pm n]$ and then, evolve the population by evolutionary improvements on the fitness function, which is obtained through application of the exact linear algorithm proposed in [3] (Step 2 in Algorithm 1). The population of signed permutations is ranked according to their fitness and $\mathcal{GA}_s$ continues (Algorithm 1). The theory of breakpoint graphs is crucial in order to built exact solutions for *SRD*. In fact, based on a thorough analysis over alternating cycles of breakpoint graphs, Hannenhalli and Pevzner showed that

$$d(\pi) = \begin{cases} b(\pi) - c(\pi) + h(\pi) + 1 \text{ or} \\ b(\pi) - c(\pi) + h(\pi) \end{cases}$$

where π is an unsigned permutation resulting from the transformation of a signed permutation and, $h(\pi)$ denotes the number of *hurdles* of π, a parameter related to the connected components of a special graph built from the breakpoint graph of π. Furthermore, $d(\pi) = b(\pi) - c(\pi) + h(\pi) + 1$ only for *fortresses*, a class of permutations classified as "hard to sort" and characterized in [17]. The linear algorithm proposed in [3] is built from the analysis of the *overlap graph*, a graph where the vertices are alternating cycles and the edges are defined from the concept of *interleaved cycles* of the breakpoint graph. Thus, breakpoint graphs are far too related with computation of fitness in our parallel GA.

Other approaches to build algorithms to solve *URD* are based on applying prioritarily reversals that reduce the maximum number of breakpoints. Notice that, if one apply a reversal ρ on a permutation π, then $|b(\pi\rho) - b(\pi)| \leq 2$, that is the number of breakpoints decreases or increases at most by two. By the relation between $b(\pi) - c(\pi)$ and $d(\pi)$, reversals that increase $c(\pi)$ or that decrease $b(\pi)$ by two are prioritized, expecting that this local optimization approach leads to a global optimization ([5], [6], [20]).

B. Genetic Algorithm for URD

The algorithm $\mathcal{GA}_s$ given in [12], is the standard GA used in our parallel approach. The input of $\mathcal{GA}_s$ is an unsigned permutation π of length n and its output a number of reversals to sort π. Initially, a population of $n \log n$ individuals that are signed permutations generated from π by randomly assigning signs to each element of π is generated. The exact solutions of these individuals are feasible solutions for π. The fitness of each one of these individuals is computed through application of the linear algorithm for solving *SRD* proposed in [3].

The genetic operators used in $\mathcal{GA}_s$ are: *selection* that considers a percentage of the best individuals in the population; and, *crossover* that uses a single-point cut mechanism. As policy of insertion of the offspring, the candidates for leaving the current population are those with the worst fitness values. To improve the population quality, in each generation two best current individuals are selected for which crossover and *mutation* operations are applied generating two new descendants. If the new offspring are better than those individuals in the population, they are incorporated in the current population. The $\mathcal{GA}_s$ finishes after n generations are completed.

The pseudo-code of $\mathcal{GA}_s$is shown in Algorithm 1.

C. Parallel Island Models GA

Island models are an efficient way of implementing bio-inspired algorithms in parallel where an island is usually represented by a processor. In this work we are considering only GAs. In this model, each island runs an instance of the GA maintaining its own population and evolving independently and, periodically, a portion of its population is interchanged in a process called migration. This process involves a set of parameters that impact both the performance and accuracy in multi-island algorithms [21]. By default, the target island is the

Algorithm 1: GA for solving URD

Input: Unsigned permutation π
Output: Approximation of the minimum number of reversals that transform π in id.

```
Generate the initial population of signed permutations;
Compute fitness of the initial population;
for i = 1 to Length(π) do
    Perform selection and save the best solution found;
    Apply the crossover operator;
    Apply the mutation operator;
    Compute the fitness of the current population;
    Perform replacement of the worst individuals;
```

one that will receive individuals from a local island. Below we highlight the required parameters.

- *NumMigIndividuals:* amount of individuals that will be sent and received from one island to another.
- *TypeEmIndividual:* type of individuals selected for emigration, which will be sent to the target island. The types of individuals are three: 1) better; 2) worse; 3) random.
- *EmPolicy:* type of emigration policy in the local island, that is whether the emigrating individuals are clones or they are indeed removed from the source and sent to the target island: 1) clone individuals; 2) remove individuals.
- *TypeImIndividual:* type of immigration policy to select the type of individuals to be replaced by immigrants to the target island. Possible types are: 1) worse; 2) random; and 3) similar individuals (i.e., same fitness).
- *MigrationInterval:* generation migration interval when islands exchange individuals.
- *Topology*: graph connectivity model that specifies which pairs of islands will communicate.

In [22], Cantú-Paz reported that Parallel island models have the capacity to provide better results than algorithms containing a single large population, both in performance and accuracy. The reason for the improvement in performance is linked to parameters as *MigrationInterval* and *Topology* as discussed by Lässig and Sudholt in [23]. For explaining the improvement in accuracy, they argue that the different islands can explore different points in the search space of possible populations, which is not possible to be performed by models containing a single population. In addition to the usual requirements of sequential GAs, parallel GAs require a careful investigation into the impact of parameter setting and specific needs are to be considered for each different problem, as pointed out by Leitão et al. in [24].

D. Communication Topologies in the Island Models

The topology is an important factor in the performance of parallel multi-island GAs since it determines how slow or fast a good solution disseminates to other islands. If the topology has dense connectivity, good solutions will quickly spread to all islands and can quickly take over the population, classic models use complete graph, 4-D Hipercube and 4×4 Toroidal mesh [22]. On the other hand, if the topology is sparse, the solutions will spread more slowly allowing the emergence of different solutions; consequently, these solutions may subsequently form potentially better individuals. Ring topologies have been successful in various works (e.g., [25], [26], [27], [28], [14]). In addition, the topology also has an impact on the cost (run-time) of migration. Densely connected topologies can promote a better mix of individuals, but also entail higher communication costs, as well as have a high probability of reaching premature convergence.

Another important factor to be considered when choosing a topology is whether the islands will be organized statically or dynamically. Static topologies are specified at the beginning of execution and remain unchanged. In the dynamic case, communication to be done between islands is not fixed, but is determined by some criterion. The motivation behind dynamic topologies is to identify situations where migrants are likely to have some effect. The criteria used to choose an island as destination include measures of population diversity [29], a measure of the genotypic distance between the islands [30] or attractiveness factor between islands [31].

III. Parallel Island Models for URD

Before presenting the new parallel island models, a quick description of the approach proposed in [14] would be given. The operation of the new models proposed here are based on the same background, since the new algorithms differ essentially in the communication topology. In [14] two models, called $\mathcal{GA}_{\text{RNP}}$ and $\mathcal{GA}_{\text{RP}}$, were proposed that are based on the unidirectional ring communication topology and other two models, called $\mathcal{GA}_{\text{CNP}}$ and $\mathcal{GA}_{\text{CP}}$, that are based on the complete graph topology. For all models, each process represents an island running an instance of $\mathcal{GA}_{\text{S}}$ with a different population of size $n \log n$. The breeding cycle is homogeneous with equal parameter set on all islands. The interval between one migration phase and another is defined as a percentage of the total number of generations in the breeding cycle. In a migration phase, each island selects an amount of individuals (parameter *NumMigIndividuals*) of a given type, as defined in parameter *TypeEmIndividual*, which can be cloned or removed (parameter *EmPolicy*), to be sent to their respective neighbor (target island). The target island inserts the immigrants into its population by replacing its own individuals according to the configuration of parameter *TypeImIndividual*. The output of all models are the best result among all islands.

The models $\mathcal{GA}_{\text{CNP}}$ and $\mathcal{GA}_{\text{RNP}}$ differ from $\mathcal{GA}_{\text{CP}}$ and $\mathcal{GA}_{\text{RP}}$ in the way in which the population is generated. For $\mathcal{GA}_{\text{CNP}}$ and $\mathcal{GA}_{\text{RNP}}$, each island generates its own population consisting of $n \log n$ individuals from the input permutation by randomly assigning a (negative or positive) sign to each of its genes. For $\mathcal{GA}_{\text{CP}}$ and $\mathcal{GA}_{\text{RP}}$, the whole population is generated and partitioned into sub-populations of equal size to be allocated on each island. Subscripts "NP" and "P" are used to discriminate between models with *non partitioned* and *partitioned* initial populations. Algorithm 2 describes how the initial population is generated by Island 0, which constructs a

Algorithm 2: Generation of the population for each island

Input: Unsigned permutation π
Output: Each island with its own initial population

```
p = numberIslands;
Island 0 generates n log n signed permutations from π for itself;
for i = 1 to p − 1 do
    Island 0 generates n log n signed permutations from π and send them to Island i (MPI_Send);
for i = 1 to p − 1 do
    Island i receives n log n signed permutations from Island 0 (MPI_Recv);
```

population of size $p * n \log n$, where p represents the number of islands, and sends populations of size $n \log n$ individuals to each other island. As in [14] the *(Message Passing Interface) MPI* library is used. The parallel commands `MPI_Send` and `MPI_Recv` are used for the exchange of messages among islands (represented by processes), and `MPI_Barrier` to perform the synchronization. Experiments in [14] showed that $\mathcal{GA}_{\text{RP}}$ computed the best results in terms of accuracy when compared to the other algorithms in the literature. Regarding the run-time performance $\mathcal{GA}_{\text{RP}}$ had showed a speed-up factor of around eleven in comparison to $\mathcal{GA}_{\text{S}}$.

IV. New Parallel Island Models GA for URD

The new models are based on static and dynamic topologies for which the breeding cycle, the migration interval and the population generationfollows as described in Section III.

New island models are proposed using bidirectional static communication topologies: torus, tree and a 4×3-net (Fig. 4). Regarding to communication balancing, interactions between nodes (representing islands) are larger in the torus topology, in which all nodes communicate with four other nodes, causing greater spread of genes through the neighborhood when compared with tree and 4×3-net topologies. In the tree topology, internal nodes communicate with at least two and at most three other nodes, whereas leave nodes with a single node. In the 4×3-net all frontier nodes are connected with three nodes except the two internals which are connected with four nodes. In essence, such topologies offer different interactions between islands, providing an environment conducive to observe how miscegenation will impact in performance and accuracy of results. In the proposed dynamic topologies a bi-directional chained lists built from a complete graph is used. In a complete graph all possible connections between islands are possible at run-time and, according to the strategy used, a bi-directional chained lists containing the links between the islands is generated, to be used during the migration process. The strategies are based on the diversity and fitness computed in each island, with such strategies it is expected to produce a much more intense miscegenation than with static topologies, since in each migration phase there is the possibility of interactions among all islands. Details of both static and dynamic models are presented in the following sections.

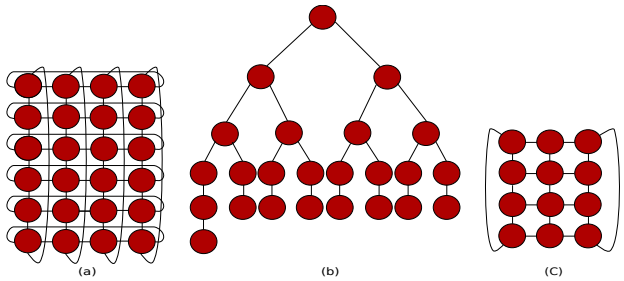

Fig. 4. Topology sketches with nodes representing islands and edges representing communication between islands: (a) Torus, (b) Tree, (c) 4×3-net.

A. Parallel GA with Static Topology

Two Parallel Island Models GA (for short, *PIMGA*) were proposed for each topology as follows:

1) *PIMGAs* with each island randomly generating its own population as described in Section III: $\mathcal{GA}_{\text{TrNP}}$ from the tree topology; $\mathcal{GA}_{\text{ToNP}}$ from the torus topology; and, $\mathcal{GA}_{\text{NeNP}}$ from the 4×3-net topology.
2) *PIMGAs* where each island's population is generated according to Algorithm 2: $\mathcal{GA}_{\text{TrP}}$ from the tree topology; $\mathcal{GA}_{\text{ToP}}$ from the torus topology; and, $\mathcal{GA}_{\text{NeP}}$ from the 4×3-net topology.

B. Parallel GA with Dynamic Topology

Initially, islands qualify as: good, bad and medium. The concept of good, bad and medium is linked to the diversity of each island. To compute the status of each island, we use two metrics: variance and average. The variance, computes the diversity in each island, high variance represents high diversity in the population. From the average, it is possible to verify the development of the island. Then, using both metrics, a ranking of the islands involved is performed. The ranking process always happens before migration starts. The proposed algorithms are presented below.

1) *PIMGAs* with each island randomly generating its own population, as described in Section III:
   - $\mathcal{GA}_{\text{GMBNP}}$, in this model the communication is restricted to be between islands with the same qualification (good, medium or bad).
   - $\mathcal{GA}_{\text{GBMMNP}}$, in this model, good islands exchange individuals with bad islands, and medium islands exchange individuals with medium islands.
   - $\mathcal{GA}_{\text{RdNP}}$, in this model the communication between islands is done randomly: let p be the number of islands; initially, we create an array of size p containing the identifier of each island, then we apply n random shifts of positions in the array. Later, pairs of indexes (containing the identifier of two islands) in the array are taken, which will share individuals among themselves, for instance: $(0,1),(2,3),\cdots,(k-2,k-1)$,

emphasizing that the indices are referring the array and not identifiers of islands.

2) *PIMGAs* where each island has the population generated according to Algorithm 2:
   - $\mathcal{GA}_{\text{GMBP}}$ with communication as for $\mathcal{GA}_{\text{GMBNP}}$.
   - $\mathcal{GA}_{\text{GBMMP}}$ with communication as for $\mathcal{GA}_{\text{GBMMNP}}$.
   - $\mathcal{GA}_{\text{RdP}}$ with communication as for $\mathcal{GA}_{\text{RdNP}}$.

V. Experiments and Results

The proposed models in Sections IV-A and IV-B were implemented using the `MPI` and `OpenMP` libraries of the `C` language. This allows parallelism in two levels. The experiments were executed on a computer with 256 GB of RAM, and two processors Xeon E5-2620 with hyper-threading, where each processor has 6 cores. Up to 24 threads were used simultaneously, which can be done without degrading the performance over this platform.

For the experiments, always performed over global populations of the same size, we divided the proposed *PIMGAs* into two classes, which differ in the number of islands.

1) *PIMGAs* with 24 islands with populations of $n \log n$ individuals: $\mathcal{GA}_{\text{TrNP}}$, $\mathcal{GA}_{\text{TrP}}$, $\mathcal{GA}_{\text{ToNP}}$, $\mathcal{GA}_{\text{ToP}}$.
2) *PIMGAs* with 12 islands with $2n \log n$ individuals: $\mathcal{GA}_{\text{NeNP}}$, $\mathcal{GA}_{\text{NeP}}$, $\mathcal{GA}_{\text{GMBNP}}$, $\mathcal{GA}_{\text{GMBP}}$, $\mathcal{GA}_{\text{GBMMNP}}$, $\mathcal{GA}_{\text{GBMMP}}$, $\mathcal{GA}_{\text{RdNP}}$, $\mathcal{GA}_{\text{RdP}}$.

In the first class, `MPI` is used to exchange individuals between islands. In the latter class, parallelism is applied at the global level as in the former class, (`MPI` is used) to exchange individuals between islands; and at the local level, each island uses `OpenMP` with two threads to accelerate the individuals' fitness computation speeding up the processing time.

In order to set up the parameters (see Table I), in such a manner that the accuracy is improved (converging to better approximations of the minimum number of reversals), several parameter combinations were exhaustively tested. Experiments to set up parameters were conducted as follows: groups of twenty permutations with n genes for $n \in \{50, 60, 70, \ldots, 150\}$ were given as input. Each algorithm proposed in Sections IV-A and IV-B was executed ten times for each permutation in each group. To adjust each parameter, different good values (Table I) were empirically estimated and random values were set for parameters that were not yet defined through the experiments. After that, the parameters values that gave the best results for each proposed algorithm were chosen (see Table II).

TABLE I
Estimated Values for the Parameters of the *PIMGAs*

Parameter	estimated values
Crossover probability	$40\%, 42\%, \cdots, 98\%, 100\%$
Mutation probability	$1\%, 2\%, \cdots, 5\%$
Percentage for selection	$40\%, 42\%, \cdots, 98\%, 100\%$
Percentage for replacement	$40\%, 42\%, \cdots, 98\%, 100\%$
NumMigIndividuals	1,2,3,4,5,6,7,8,9,10,11,12,13
TypeEmIndividual	1=Better, 2=Worse, 3=Random
EmPolicy	1=Clone, 2=Remove
TypeImIndividual	1=Worse, 2=Random, 3=Similar
MigrationInterval	$1\%, 2\%, \cdots, 99\%, 100\%$

A. Experiments with Synthetic Permutations

Two experiments were performed. The first one computes the speed-up of the proposed *PIMGAs* regarding $\mathcal{GA}_{\text{S}}$ and, the second one the accuracy regarding $\mathcal{GA}_{\text{RP}}$, which as previously mentioned is the best known algorithm found in the literature for solving the URD problem.

In the first experiment, one hundred permutations of size 150 were randomly generated, and then $\mathcal{GA}_{\text{S}}$, $\mathcal{GA}_{\text{RP}}$ and the proposed *PIMGAs* were executed ten times for each input. The run-time for each input and algorithm was taken as the mean time of these executions. Average results are presented in Table III.

In the second experiment, groups of one hundred permutations of size 50 to 150 were randomly generated. Algorithm $\mathcal{GA}_{\text{RP}}$ and the proposed *PIMGAs* were executed ten times for each permutation in each group and the mean number of reversals computed was taken as result for each input and algorithm. The results are shown in Table IV where the value in each cell represents the average reversal distance computed for each package of one hundred permutations.The best results are highlighted in bold font.

Table III shows that the algorithm $\mathcal{GA}_{\text{GBMMNP}}$ has the best speed-up (16.02), followed by the algorithm $\mathcal{GA}_{\text{ToNP}}$ (15.93). Table IV shows that the algorithm $\mathcal{GA}_{\text{RdP}}$ has the best results in 8 cases out of 11, and also for the harder cases that are for permutations of length greater that 120.

VI. Discussion

In order to validate the results shown on Table IV, a statistical analysis was performed as proposed by Demšar [32] (and, as done in [14]). The statistical analysis was performed taking as samples the results of each algorithm for one hundred permutations of a given length. Additionally, the multiplicative inverse of each element of the samples was calculated in order to compare performances. The required steps for the statistical comparison of the algorithms are given below.

- The Friedman test is used to test the null hypothesis that all algorithms have the same performance.
- If the null hypothesis of the Friedman test is rejected, then a post-hoc test is performed; in this case, the Holm test. The Holm test is used to test the null hypothesis that a control algorithm has the same performance regarding each of the remaining algorithms.

The significance level used for both the Friedman and Holm tests is $\alpha = 0.05$ which is established as default by their corresponding implementations in JAVA (`CONTROL TEST` package), available at `sci2s.ugr.es/sicidm`.

The Friedman test rejected the null hypothesis (p-value $\leq \alpha$) that all algorithms have the same performance just for samples corresponding to the results of the algorithms for permutations of length greater than or equal to 70; then the Holm test was performed for these cases. Tables V and VI show the results of the Holm test, where the p-values in bold are those cases

TABLE II

PARAMETER SETTINGS FOR THE *PIMGAs*. 1=$\mathcal{GA}_{\text{RP}}$, 2=$\mathcal{GA}_{\text{RdNP}}$, 3=$\mathcal{GA}_{\text{RdP}}$, 4=$\mathcal{GA}_{\text{GMBNP}}$, 5=$\mathcal{GA}_{\text{GMBP}}$, 6=$\mathcal{GA}_{\text{GBMMNP}}$, 7=$\mathcal{GA}_{\text{GBMMP}}$, 8=$\mathcal{GA}_{\text{NeNP}}$, 9=$\mathcal{GA}_{\text{NeP}}$, 10=$\mathcal{GA}_{\text{ToNP}}$, 11=$\mathcal{GA}_{\text{ToP}}$, 12=$\mathcal{GA}_{\text{TrNP}}$, 13=$\mathcal{GA}_{\text{TrP}}$.

Parameter	1	2	3	4	5	6	7	8	9	10	11	12	13
Crossover probability	96%	97%	96%	96%	92%	97%	96%	92%	92%	90%	88%	96%	94%
Mutation probability	1%	1%	1%	1%	1%	1%	1%	1.4%	1%	1%	1%	1%	1%
Perc. for selection	90%	96%	94%	84%	86%	78%	94%	80%	92%	92%	96%	97%	94%
Perc. for replacement	40%	62%	50%	60%	60%	50%	54%	40%	62%	62%	62%	50%	50%
NumMigIndividuals	3	6	4	7	10	2	5	2	1	3	1	3	2
TypeEmIndividual	2	1	3	2	3	3	3	1	2	2	2	1	3
EmPolicy	1	2	2	1	1	1	2	2	1	2	2	2	1
TypeImIndividual	1	1	1	2	1	2	3	1	1	2	1	2	2
MigrationInterval	50%	88%	84%	20%	60%	30%	68%	10%	20%	94%	64%	55%	10%

TABLE III

SPEED-UP FOR THE EXPERIMENT WITH PERMUTATIONS OF SIZE 150. 1=$\mathcal{GA}_{\text{RP}}$, 2=$\mathcal{GA}_{\text{RdNP}}$, 3=$\mathcal{GA}_{\text{RdP}}$, 4=$\mathcal{GA}_{\text{GMBNP}}$, 5=$\mathcal{GA}_{\text{GMBP}}$, 6=$\mathcal{GA}_{\text{GBMMNP}}$, 7=$\mathcal{GA}_{\text{GBMMP}}$, 8=$\mathcal{GA}_{\text{NeNP}}$, 9=$\mathcal{GA}_{\text{NeP}}$, 10=$\mathcal{GA}_{\text{ToNP}}$, 11=$\mathcal{GA}_{\text{ToP}}$, 12=$\mathcal{GA}_{\text{TrNP}}$, 13=$\mathcal{GA}_{\text{TrP}}$.

Length	1	2	3	4	5	6	7	8	9	10	11	12	13
150	10.37	12.90	10.38	14.52	11.12	16.02	13.17	14.42	11.78	15.93	10.54	13.87	9.89

TABLE IV

AVERAGE RESULTS (A) FOR THE EXPERIMENT WITH HUNDRED SYNTHETIC PERMUTATIONS. 1=$\mathcal{GA}_{\text{RP}}$, 2=$\mathcal{GA}_{\text{RdNP}}$, 3=$\mathcal{GA}_{\text{RdP}}$, 4=$\mathcal{GA}_{\text{GMBNP}}$, 5=$\mathcal{GA}_{\text{GMBP}}$, 6=$\mathcal{GA}_{\text{GBMMNP}}$, 7=$\mathcal{GA}_{\text{GBMMP}}$, 8=$\mathcal{GA}_{\text{NeNP}}$, 9=$\mathcal{GA}_{\text{NeP}}$, 10=$\mathcal{GA}_{\text{ToNP}}$, 11=$\mathcal{GA}_{\text{ToP}}$, 12=$\mathcal{GA}_{\text{TrNP}}$, 13=$\mathcal{GA}_{\text{TrP}}$.

	1	2	3	4	5	6	7	8	9	10	11	12	13
L	Average												
50	36.37	36.38	36.37	36.37	36.37	36.4	36.39	**36.36**	36.37	36.39	36.38	36.38	36.38
60	44.56	44.58	**44.54**	44.57	44.55	44.59	44.57	**44.54**	**44.54**	44.57	44.56	44.56	44.56
70	52.7	52.7	52.68	52.68	52.7	52.74	52.72	**52.64**	52.66	52.74	52.74	52.7	52.72
80	61.1	61.06	**61.02**	61.06	61.04	61.1	61.1	61.03	61.05	61.16	61.13	61.07	61.1
90	69.55	69.49	**69.36**	69.44	69.44	69.53	69.5	69.41	69.4	69.56	69.59	69.49	69.52
100	77.64	77.59	**77.48**	77.56	77.55	77.69	77.63	77.51	77.53	77.74	77.72	77.59	77.64
110	86.29	86.19	86.11	86.13	86.14	86.32	86.22	**86.1**	86.14	86.35	86.36	86.26	86.28
120	94.87	94.76	**94.62**	94.72	94.7	94.88	94.82	94.67	94.67	94.94	94.94	94.85	94.84
130	103.61	103.49	**103.27**	103.38	103.34	103.64	103.54	103.37	103.39	103.7	103.72	103.57	103.59
140	112.13	112.01	**111.84**	111.93	111.92	112.19	112.02	111.89	111.88	112.22	112.2	112.08	112.11
150	120.91	120.82	**120.66**	120.68	120.73	121.03	120.83	120.69	120.7	121.03	121.05	120.89	120.89

for which the null hypothesis was rejected (p-value $\leq \alpha/i$), where i corresponds to the ith algorithm ($1 \leq i \leq 12$).

It can be observed (Tables V and VI) that the control algorithm for all cases is $\mathcal{GA}_{\text{RdP}}$, which means it is the one with best performance (minimum rank). Note that $\mathcal{GA}_{\text{RdP}}$ has statistically significant difference with respect to most of the remaining algorithms (where p-values are in bold). Also, $\mathcal{GA}_{\text{RdP}}$ does not have statistically significant difference with respect to: $\mathcal{GA}_{\text{GMBP}}$ in 6 cases out of 9; $\mathcal{GA}_{\text{NeNP}}$ in 4 cases out of 9; and $\mathcal{GA}_{\text{GMBNP}}$ in 4 cases out of 9.

VII. CONCLUSIONS AND FUTURE WORK

New parallel island models GA were proposed to solve *URD*. These models use three static and one dynamic communication topologies. The static topologies were tree, torus, and a 4×3-net, and for each one, variants were proposed in which either each island generates its own population or the whole population is generated and then split among all islands ($\mathcal{GA}_{\text{TrNP}}$, $\mathcal{GA}_{\text{ToNP}}$, $\mathcal{GA}_{\text{NeNP}}$, and $\mathcal{GA}_{\text{TrP}}$, $\mathcal{GA}_{\text{ToP}}$, $\mathcal{GA}_{\text{NeP}}$).

The models with dynamic topology use bi-directional chained lists built from a complete graph, and variants were proposed considering diversity in each island, for guiding the migration process. Also, two categories were developed according to generation of populations for each island by the own island or split from a general population ($\mathcal{GA}_{\text{GMBNP}}$, $\mathcal{GA}_{\text{GBMMNP}}$, $\mathcal{GA}_{\text{RdNP}}$, and $\mathcal{GA}_{\text{GMBP}}$, $\mathcal{GA}_{\text{GBMMP}}$, $\mathcal{GA}_{\text{RdP}}$).

An experiment was performed using packages of randomly generated permutations of sizes from 50 to 150. It was observed that $\mathcal{GA}_{\text{RdP}}$ computes the best results in terms of accuracy for most of the inputs. Algorithm $\mathcal{GA}_{\text{RdP}}$ uses the dynamic topology, a migration policy that selects immigrants and emigrants randomly, and the population of each island is taken from a global generated populations. Notice that this model promoves diversity, in contrast to others in which for instance, as migration policy good individuals are selected to replace bad individuals. The significance of these observations were confirmed by the Holm test.

A second experiment using packages of permutations of size 150, was performed for measuring the speed-up of the proposed *PIMGAs* regarding the standard sequential GA ($\mathcal{GA}_{\text{s}}$ in [12]). This experiment confirmed that $\mathcal{GA}_{\text{GBMMNP}}$ has a the best speed-up (16.02) and that $\mathcal{GA}_{\text{RdP}}$, that is the best parallel

TABLE V
RESULTS OF THE HOLM TEST FOR SETS OF HUNDREDS SYNTHETIC PERMUTATIONS WITH LENGTHS FROM 70 TO 100.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
70	$\mathcal{GA}_{\text{RdP}}$ (Rank: 6.11)	12	$\mathcal{GA}_{\text{GBMMNP}}$	8.03	**4.90098E-4**	0.00417
		11	$\mathcal{GA}_{\text{ToP}}$	7.965	**7.56905E-4**	0.00455
		10	$\mathcal{GA}_{\text{ToNP}}$	7.87	**0.0014**	0.005
		9	$\mathcal{GA}_{\text{GBMMP}}$	7.265	0.02494	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	7.345	0.03598	0.00625
		7	$\mathcal{GA}_{\text{NeNP}}$	7.04	0.0913	0.00714
		6	$\mathcal{GA}_{\text{GMBP}}$	6.92	0.14137	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.825	0.19421	0.01
		4	$\mathcal{GA}_{\text{RP}}$	6.725	0.26415	0.0125
		3	$\mathcal{GA}_{\text{TrNP}}$	6.49	0.49022	0.01667
		2	$\mathcal{GA}_{\text{GMBNP}}$	6.285	0.75068	0.025
		1	$\mathcal{GA}_{\text{NeP}}$	6.13	0.97103	0.05
80	$\mathcal{GA}_{\text{RdP}}$ (Rank: 5.265)	12	$\mathcal{GA}_{\text{ToNP}}$	8.755	**2.34689E-10**	0.00417
		11	$\mathcal{GA}_{\text{ToP}}$	8.07	**3.52461E-7**	0.00455
		10	$\mathcal{GA}_{\text{RP}}$	7.560	**3.08646E-5**	0.005
		9	$\mathcal{GA}_{\text{GBMMP}}$	7.515	**4.40226E-5**	0.00556
		8	$\mathcal{GA}_{\text{GBMMNP}}$	7.43	**8.46099E-5**	0.00625
		7	$\mathcal{GA}_{\text{TrP}}$	7.420	**9.12336E-5**	0.00714
		6	$\mathcal{GA}_{\text{RdNP}}$	6.745	**0.00721**	0.00833
		5	$\mathcal{GA}_{\text{TrNP}}$	6.66	0.01131	0.01
		4	$\mathcal{GA}_{\text{NeNP}}$	6.65	**0.01191**	0.0125
		3	$\mathcal{GA}_{\text{NeP}}$	6.515	0.02323	0.01667
		2	$\mathcal{GA}_{\text{GMBNP}}$	6.51	**0.02379**	0.025
		1	$\mathcal{GA}_{\text{GMBP}}$	5.905	0.24522	0.05
90	$\mathcal{GA}_{\text{RdP}}$ (Rank: 4.465)	12	$\mathcal{GA}_{\text{ToP}}$	9.095	**4.2203E-17**	0.00417
		11	$\mathcal{GA}_{\text{RP}}$	8.365	**1.42951E-12**	0.00455
		10	$\mathcal{GA}_{\text{ToNP}}$	8.33	**2.25691E-12**	0.005
		9	$\mathcal{GA}_{\text{GBMMNP}}$	8.17	**1.73073E-11**	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	7.795	**1.48281E-9**	0.00625
		7	$\mathcal{GA}_{\text{TrNP}}$	7.25	**4.26627E-7**	0.00714
		6	$\mathcal{GA}_{\text{GBMMP}}$	7.115	**1.49758E-6**	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.905	**9.41171E-6**	0.01
		4	$\mathcal{GA}_{\text{NeP}}$	6.655	**6.99826E-5**	0.0125
		3	$\mathcal{GA}_{\text{GMBP}}$	5.925	**0.00803**	0.01667
		2	$\mathcal{GA}_{\text{GMBNP}}$	5.65	0.03143	0.025
		1	$\mathcal{GA}_{\text{NeNP}}$	5.28	0.13893	0.05
100	$\mathcal{GA}_{\text{RdP}}$ (Rank: 4.425)	12	$\mathcal{GA}_{\text{ToNP}}$	9.49	**3.70031E-20**	0.00417
		11	$\mathcal{GA}_{\text{ToP}}$	9.125	**1.4173E-17**	0.00455
		10	$\mathcal{GA}_{\text{GBMMNP}}$	8.58	**4.55208E-14**	0.005
		9	$\mathcal{GA}_{\text{RP}}$	7.505	**2.24078E-8**	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	7.335	**1.26645E-7**	0.00625
		7	$\mathcal{GA}_{\text{GBMMP}}$	7.105	**1.13863E-6**	0.00714
		6	$\mathcal{GA}_{\text{TrNP}}$	6.63	**6.23916E-5**	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.495	**1.7096E-4**	0.01
		4	$\mathcal{GA}_{\text{NeNP}}$	6.325	**5.61008E-4**	0.0125
		3	$\mathcal{GA}_{\text{NeP}}$	6.26	**8.62932E-4**	0.01667
		2	$\mathcal{GA}_{\text{GMBNP}}$	5.94	**0.00595**	0.025
		1	$\mathcal{GA}_{\text{GMBP}}$	5.785	**0.01354**	0.05

island model regarding accuracy, has also a competitive speed-up (10.38) as well as all other proposed *PIMGAs*.

Also, it is important to stress that experiments also show that in most of the cases the new approaches outperform the previous best known parallel GA ($\mathcal{GA}_{\text{RP}}$ in [14]) in performance as well as in accuracy.

As future work, we are planning the development of heterogeneous island models in which each island might perform a different EA or explore different parameters as well as different migration policies than the other islands.

REFERENCES

[1] S. Hannenhalli and P. Pevzner, "Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals," in Proc. of the 27^{th} annual ACM Symposium on Theory of Computing, 1995, pp. 178–189.

[2] P. Berman and S. Hannenhalli, "Fast sorting by reversal," in Proc. Combinatorial Pattern Matching (CPM). Springer, 1996, pp. 168–185.

[3] D. A. Bader, B. M. Moret, and M. Yan, "A linear-time algorithm for computing inversion distance between signed permutations with an experimental study," J. of Comp. Biol., vol. 8, no. 5, pp. 483–491, 2001.

[4] A. Caprara, "Sorting by reversals is difficult," in Proc. of the first annual Int. Conf. on Comp. molecular Biology. ACM, 1997, pp. 75–83.

[5] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for the inversion distance between two chromosomes," in Proc. Combinatorial Pattern Matching (CPM). Springer, 1993, pp. 87–105.

[6] V. Bafna and P. A. Pevzner, "Genome rearrangements and sorting by reversals," SIAM J. on Computing, vol. 25, no. 2, pp. 272–289, 1996.

[7] D. A. Christie, "A 3/2-approximation algorithm for sorting by reversals," in Proc. of the ninth annual ACM-SIAM symposium on Discrete algorithms. Soc. for Ind. and App. Mathematics, 1998, pp. 244–252.

[8] P. Berman, S. Hannenhalli, and M. Karpinski, "1.375-approximation algorithm for sorting by reversals," in Proc. Algorithms ESA. Springer, 2002, pp. 200–210.

[9] A. Auyeung and A. Abraham, "Estimating genome reversal distance by genetic algorithm," in Congress on Evolutionary Computation (CEC), vol. 2. IEEE, 2003, pp. 1157–1161.

[10] A. Ghaffarizadeh, K. Ahmadi, and N. S. Flann, "Sorting unsigned permutations by reversals using multi-objective evolutionary algorithms with variable size individuals," in 2011 IEEE Congress of Evolutionary Computation (CEC), June 2011, pp. 292–295.

[11] J. L. Soncco-Álvarez and M. Ayala-Rincón, "A genetic approach with a simple fitness function for sorting unsigned permutations by reversals," in Proc. Colombian Computing Congress (CCC). IEEE, 2012, pp. 1–6.

[12] ——, "Sorting permutations by reversals through a hybrid genetic algorithm based on breakpoint elimination and exact solutions for signed permutations," Elec. Notes in Theor. C.S., vol. 292, pp. 119–133, 2013.

[13] J. L. Soncco-Álvarez, G. M. Almeida, J. Becker, and M. Ayala-Rincón, "Parallelization and virtualization of genetic algorithms for sorting permutations by reversals," in World Congress on Nature and Biologically Inspired Computing (NaBIC). IEEE, 2013, pp. 29–35.

[14] L. A. da Silveira, J. L. Soncco-Álvarez, and M. Ayala-Rincón, "Parallel genetic algorithms with sharing of individuals for sorting unsigned genomes by reversals," in 2017 IEEE Congress on Evolutionary Computation (CEC), June 2017, pp. 741–748.

[15] T. A. de Lima and M. Ayala-Rincón, "On the average number of reversals needed to sort signed permutations," Discrete Applied Mathematics, vol. 235, no. Supplement C, pp. 59 – 80, 2018.

[16] A. Caprara, "On the tightness of the alternating-cycle lower bound for sorting by reversals," J. Comb. Optim., vol. 3, no. 2-3, pp. 149–182, 1999. [Online]. Available: https://doi.org/10.1023/A:1009838309166

[17] S. Hannenhalli and P. A. Pevzner, "Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals," J. of the ACM, vol. 46, no. 1, pp. 1–27, Jan. 1999.

[18] J. L. Soncco-Álvarez, G. M. Almeida, J. Becker, and M. Ayala-Rincón, "Parallelization of genetic algorithms for sorting permutations by reversals over biological data," Int. Journal of Hybrid Intelligent Systems, vol. 12, no. 1, pp. 53–64, 2015.

[19] J. L. Soncco-Álvarez, D. M. Muñoz, and M. Ayala-Rincón, "Opposition-Based Memetic Algorithm and Hybrid Approach for Sorting Permutations by Reversals," Evolutionary Computation, 2018, accepted.

[20] D. A. Christie, "A 3/2-approximation algorithm for sorting by reversals." in SODA, 1998, pp. 244–252.

[21] D. Sudholt, Springer Handbook of Computational Intelligence. Springer, 2015, ch. Parallel Evolutionary Algorithms, pp. 929–959.

[22] E. Cantú-Paz, "A survey of parallel genetic algorithms," Calculateurs Paral., Reseaux et Sys. Repartis, vol. 10, no. 2, pp. 141–171, 1998.

[23] J. Lässig and D. Sudholt, "Design and analysis of migration in parallel evolutionary algorithms," Soft Computing, vol. 17, no. 7, pp. 1121–1144, Jul 2013.

[24] A. Leitão, F. B. Pereira, and P. Machado, "Island models for cluster geometry optimization: how design options impact effectiveness and diversity," J. of Global Opt., vol. 63, no. 4, pp. 677–707, Dec 2015.

[25] T. C. Belding, "The distributed genetic algorithm revisited," in Proc. of the 6th Int. Conf. on Genetic Algorithms. Morgan Kaufmann Publishers Inc., 1995, pp. 114–121.

[26] D. Whitley, S. Rana, and R. B. Heckendorn, "The island model genetic algorithm: On separability, population size and convergence," CIT. J. of computing and inf. tech., vol. 7, no. 1, pp. 33–47, 1999.

[27] E. Alba and J. M. Troya, "Influence of the migration policy in parallel distributed gas with structured and panmictic populations," Applied Intelligence, vol. 12, no. 3, pp. 163–181, May 2000.

[28] T.-P. Hong, W.-Y. Lin, S.-M. Liu, and J.-H. Lin, "Experimental analysis of dynamic migration intervals on 0/1 knapsack problems," in 2007 IEEE Congress on Evolutionary Computation, Sept 2007, pp. 1163–1167.

[29] M. Munetomo, Y. Takai, and Y. Sato, "An efficient migration scheme for subpopulation-based asynchronously parallel genetic algorithms," in Proc. of the 5th Int. Conf. on Genetic Algorithms. Morgan Kaufmann Publishers Inc., 1993, pp. 649–.

[30] S.-C. Lin, W. F. Punch, and E. D. Goodman, "Coarse-grain parallel genetic algorithms: categorization and new approach," in Proc. of 1994 6th IEEE Symp. on Paral. and Dist. Processing, Oct 1994, pp. 28–37.

[31] G. Duarte, A. Lemonge, and L. Goliatt, "A dynamic migration policy to the island model," in 2017 IEEE Congress on Evolutionary Computation (CEC), June 2017, pp. 1135–1142.

[32] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006.

TABLE VI
RESULTS OF THE HOLM TEST FOR SETS OF HUNDREDS SYNTHETIC PERMUTATIONS WITH LENGTHS FROM 110 TO 150.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
110	$\mathcal{GA}_{\text{RdP}}$ (Rank: 4.795)	12	$\mathcal{GA}_{\text{ToP}}$	9.185	**1.57591E-15**	0.00417
		11	$\mathcal{GA}_{\text{ToNP}}$	9.125	**3.78333E-15**	0.00455
		10	$\mathcal{GA}_{\text{GBMMNP}}$	8.745	**7.3939E-13**	0.005
		9	$\mathcal{GA}_{\text{RP}}$	8.38	**7.5541E-11**	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	7.83	**3.57619E-8**	0.00625
		7	$\mathcal{GA}_{\text{TrNP}}$	7.595	**3.69741E-7**	0.00714
		6	$\mathcal{GA}_{\text{GBMMP}}$	6.825	**2.27953E-4**	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.185	0.01161	0.01
		4	$\mathcal{GA}_{\text{NeP}}$	6.14	0.0146	0.0125
		3	$\mathcal{GA}_{\text{GMBP}}$	5.525	0.18502	0.01667
		2	$\mathcal{GA}_{\text{NeNP}}$	5.44	0.24155	0.025
		1	$\mathcal{GA}_{\text{GMBNP}}$	5.23	0.42963	0.05
120	$\mathcal{GA}_{\text{RdP}}$ (Rank: 4.125)	12	$\mathcal{GA}_{\text{ToP}}$	9.255	**1.22584E-20**	0.00417
		11	$\mathcal{GA}_{\text{ToNP}}$	9.155	**6.66999E-20**	0.00455
		10	$\mathcal{GA}_{\text{GBMMNP}}$	8.33	**2.25916E-14**	0.005
		9	$\mathcal{GA}_{\text{TrNP}}$	8.085	**6.47424E-13**	0.00556
		8	$\mathcal{GA}_{\text{RP}}$	7.96	**3.32764E-12**	0.00625
		7	$\mathcal{GA}_{\text{TrP}}$	7.645	**1.64586E-10**	0.00714
		6	$\mathcal{GA}_{\text{GBMMP}}$	7.31	**7.34023E-9**	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.365	**4.75959E-5**	0.01
		4	$\mathcal{GA}_{\text{NeP}}$	6.105	**3.24327E-4**	0.0125
		3	$\mathcal{GA}_{\text{NeNP}}$	5.69	**0.00449**	0.01667
		2	$\mathcal{GA}_{\text{GMBNP}}$	5.615	**0.00682**	0.025
		1	$\mathcal{GA}_{\text{GMBP}}$	5.36	**0.02494**	0.05
130	$\mathcal{GA}_{\text{RdP}}$ (Rank: 3.4)	12	$\mathcal{GA}_{\text{ToP}}$	10.075	**8.30815E-34**	0.00417
		11	$\mathcal{GA}_{\text{ToNP}}$	9.57	**3.95218E-29**	0.00455
		10	$\mathcal{GA}_{\text{GBMMNP}}$	8.93	**1.0093E-23**	0.005
		9	$\mathcal{GA}_{\text{RP}}$	8.525	**1.33522E-20**	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	8.01	**5.74733E-17**	0.00625
		7	$\mathcal{GA}_{\text{TrNP}}$	7.84	**7.52833E-16**	0.00714
		6	$\mathcal{GA}_{\text{GBMMP}}$	7.37	**5.66714E-13**	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.485	**2.12651E-8**	0.01
		4	$\mathcal{GA}_{\text{NeNP}}$	5.775	**1.61604E-5**	0.0125
		3	$\mathcal{GA}_{\text{NeP}}$	5.665	**3.91362E-5**	0.01667
		2	$\mathcal{GA}_{\text{GMBNP}}$	4.96	**0.00462**	0.025
		1	$\mathcal{GA}_{\text{GMBP}}$	4.395	0.07082	0.05
140	$\mathcal{GA}_{\text{RdP}}$ (Rank: 4.115)	12	$\mathcal{GA}_{\text{ToNP}}$	9.44	**4.10309E-22**	0.00417
		11	$\mathcal{GA}_{\text{ToP}}$	9.125	**9.32298E-20**	0.00455
		10	$\mathcal{GA}_{\text{GBMMNP}}$	9.095	**1.53691E-19**	0.005
		9	$\mathcal{GA}_{\text{RP}}$	8.24	**6.90361E-14**	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	8.06	**7.90065E-13**	0.00625
		7	$\mathcal{GA}_{\text{TrNP}}$	7.505	**7.49998E-10**	0.00714
		6	$\mathcal{GA}_{\text{RdNP}}$	6.75	**1.71564E-6**	0.00833
		5	$\mathcal{GA}_{\text{GBMMP}}$	6.75	**1.71564E-6**	0.01
		4	$\mathcal{GA}_{\text{NeP}}$	5.945	**8.9151E-4**	0.0125
		3	$\mathcal{GA}_{\text{GMBNP}}$	5.65	**0.00532**	0.01667
		2	$\mathcal{GA}_{\text{NeNP}}$	5.325	0.02802	0.025
		1	$\mathcal{GA}_{\text{GMBP}}$	5.0	0.10808	0.05
150	$\mathcal{GA}_{\text{RdP}}$ (Rank: 4.48)	12	$\mathcal{GA}_{\text{ToP}}$	9.715	**1.99859E-21**	0.00417
		11	$\mathcal{GA}_{\text{GBMMNP}}$	9.485	**1.0135E-19**	0.00455
		10	$\mathcal{GA}_{\text{ToNP}}$	9.235	**5.94659E-18**	0.005
		9	$\mathcal{GA}_{\text{RP}}$	8.035	**1.08397E-10**	0.00556
		8	$\mathcal{GA}_{\text{TrP}}$	7.67	**6.95418E-9**	0.00625
		7	$\mathcal{GA}_{\text{TrNP}}$	7.235	**5.66781E-7**	0.00714
		6	$\mathcal{GA}_{\text{GBMMP}}$	6.605	**1.14167E-4**	0.00833
		5	$\mathcal{GA}_{\text{RdNP}}$	6.465	**3.132E-4**	0.01
		4	$\mathcal{GA}_{\text{NeNP}}$	6.145	**0.0025**	0.0125
		3	$\mathcal{GA}_{\text{NeP}}$	5.63	0.03679	0.01667
		2	$\mathcal{GA}_{\text{GMBP}}$	5.455	0.07668	0.025
		1	$\mathcal{GA}_{\text{GMBNP}}$	4.845	0.50751	0.05