Behavior of Bioinspired Algorithms in Parallel Island Models

Lucas A. da Silveira[†], José L. Soncco-Álvarez[*], Thaynara A. de Lima[**] and Mauricio Ayala-Rincón[†,‡]
Departments of [†]Computer Science and [‡]Mathematics, Universidade de Brasília, Brasília D.F., Brazil
[*]Department of Informatics, Universidad Nacional de San Antonio Abad del Cusco, Cusco, Peru
[**]Institute of Mathematics and Statistics, Universidade Federal de Goiás, Goiânia, Brazil

***Abstract*—Parallel island models are used to increase accuracy and performance (speed-up) of metaheuristics. Such models provide gains by the exchange of information between islands through the migratory process. The key to obtaining gains with parallel island models is the manipulation of migration parameters, since depending on how these parameters are handled the gains vary. Based on this assumption, this work uses three metaheuristics: genetic algorithm, self-adjusting particle swarm optimization and social spider algorithm. From each metaheuristic, parallel island models were proposed, diversifying the number of natives on the islands, and the behavior of these models were studied. The assessment confirmed the impact of variations migration parameters on accuracy and performance as well as the importance on the number of natives located on the islands. The best solutions were obtained with island models from genetic algorithm and self-adjusting particle swarm optimization, and the best speedups were achieved with island models from social spider algorithm.**

I. Introduction

Several $\mathcal{NP}$ problems cannot be satisfactorily solved in polynomial time. Exploring the vast search space for instances of such problems is still challenging, given that in terms of memory and processing power, today's computers lack sufficient resources. If the resolution to these problems is still not optimal, applying approximate methods remains a good strategy. Metaheuristics are a well-known efficient method that are applied in situations where the problems can be modeled as optimization functions using a combination of random choices and historical results to explore the problem search space.

Another way to explore vast search spaces in complex problems is to use the parallel island model (PIM) paradigm [1]. PIM is inspired by the theory of punctuated equilibrium introduced in [2], working with a global population subdivided into sub populations, called in the literature as islands, where each island is mapped in a processor that occasionally communicates with other island through the so called migration process. As an evolutionary engine, PIMs use metaheuristics based on: populations, particle swarms and colony optimization, allowing control for premature convergence, avoiding optimal locations and exploring diversity in the search space in order to increment performance and accuracy of results. Research in this field generally analyzes the role of migration rates, the number of islands and the nature of the communication network (connectivity and communication topology) in the optimization process aiming at performance and accuracy.

This work proposes and analyzes the behavior of different PIMs for three metaheuristics: Genetic algorithm ($\mathcal{GA}$), self-adjusting Particle Swarm Optimization ($\mathcal{PSO}$) and Social Spider Algoritm ($\mathcal{SSA}$). The $\mathcal{GA}$ is an evolutionary algorithm that uses a probabilistic solution search engine based on biological evolution, which combines aspects of genetic mechanics and natural selection of individuals, but that involves a high computational cost. On the other hand, we have self-adjusting $\mathcal{PSO}$ a metaheuristic based on patterns of nature that presents relatively less computational complexity w.r.t. $\mathcal{GA}$. Regarding accuracy, the $\mathcal{PSO}$ provides competitive optimization solutions for many practical problems. $\mathcal{SSA}$ is a new metaheuristic proposed for solving global optimization problems that is based on the social spider foraging strategy, using spider web vibrations to determine prey positions. Experiments comparing $\mathcal{SSA}$ with meta-statistics known in the literature showed that $\mathcal{SSA}$ provides competitive solutions for many benchmarks [3]. From the proposed PIMs, the adequability of migration policies is verified through the analysis of performance and accuracy, such that the proposed algorithms are submitted to resolution of the case-study of *unsigned reversal distance* (URD) between genomes, which is a well-known $\mathcal{NP}$-hard problem [4], [5]. For this case-study, PIMs are proposed with 12 and 24 islands using different communication topologies. The topologies are responsible for communication between islands during the migratory process.

To exploit the potential of PIMs, parameters regarding breeding cycle, collective behavior, and migration policy are set through exhaustive tests. In the experiments phase, several input sets are used for the case-study. The feedback obtained from the experiments does not point to a generic model that offers better performance and/or better accuracy; however, the results show that besides the parameters mentioned above, another important mechanism to be explored is the size of the search space allocated on each island. Whereas, PIMs for the same case-study using larger search space present superior performance and accuracy [6], [7], [8].

Initially, Sec. II presents the case-study, and concepts about PIM, $\mathcal{GA}$, $\mathcal{PSO}$ and $\mathcal{SSA}$; Sec. III discusses related work. Afterwards, Sec. IV introduces the proposed PIMs, Sec. V presents experiments, and Section VI discusses them. Finally, Sec. VII concludes and proposes future work. The source code of the proposed PIMs and an extended version that includes the statistical test are available at http://genoma.cic.unb.br.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. Background

This section presents the case-study and theoretical content regarding metaheuristics $\mathcal{GA}$, $\mathcal{PSO}$, $\mathcal{SSA}$ and research advances achieved from the parallel island models paradigm.

A. Case-study

Reversals are considered to calculate the evolutionary distance between organisms containing a single chromosome. Computing the minimum number of reversals for transforming one genome into another is known as the *reversal distance problem* [9], [5].

The genes are represented in a genome as a permutation $\pi = (\pi_1, \pi_2, ..., \pi_n)$ into the set $\{1, ..., n\}$, with n being the number of genes. A permutation of length n is a bijection from $\{1, \cdots, n\}$ into itself. The reversal operation denoted as $\rho^{j,k}$, for $1 \leq j \leq k \leq n$, inverts contiguous elements between π_j and π_k transforming π into $\pi' = (\cdots, \pi_{j-1}, \pi_k, \cdots, \pi_j, \pi_{k+1}, \cdots)$; for example: for $\pi = (1, 9, 4, 3, 5, 8, 2, 7, 6, 10)$, $\rho^{2,9}(\pi)$ gives $\pi' = (1, 6, 7, 2, 8, 5, 3, 4, 9, 10)$. There are two different types of permutations: signed and unsigned. When the orientation (positive or negative) of the genes inside the genome is considered, the genome is considered a signed permutation in which each gene has a positive or negative sign according to its orientation within the genome. Computing the minimum number of reversal between two signed permutations is known in the literature as the signed reversal distance problem and was proved to belong to $\mathcal{P}$ [10]. On the other hand, if the genes in the genome are abstracted without orientation, the representation of the genome corresponds to an unsigned permutation in which each symbol is associated with a gene. Calculating the minimum reversal distance between two unsigned permutations is called the unsigned reversal distance (URD) problem and is known to be an $\mathcal{NP}$-hard problem [4]. In genetic algorithms to deal with URD, the fitness of a unsigned genome is computed as the reversal distance of signed genomes obtained by orienting the genes in the genome as given in [11], [12]. The URD problem is the case-study considered in this work.

B. Parallel Island Model

PIM is an alternative to increase performance and accuracy of evolutionary and bioinspired algorithms in parallel architectures. Islands maintain a partial possible solutions that are enhanced by an algorithm in a parallel architecture such as multi-core computers or clusters. In the breeding cycle, islands can evolve their populations homogeneously using the same algorithm keeping the same configurations. On the other hand, if islands may execute different algorithms or change parameter settings, the model is called heterogeneous.

The evolutionary cycle is important to improve the genetic potential of island populations, but the differential of the paradigm is the migration process. In migration, periodically, individuals are exchanged between neighboring islands. These movements can have positive or negative impacts at the end of the evolutionary process, due to the variation of the genetic data on each island. Migration implies a set of decisions that need to be discussed and give rise to the choice of parameters mentioned below, as argued in [13], [14].

- *Number of Islands*: establishes the number of islands. The total natives in all the islands should be the same as the number of individuals in the sequential model.
- *Topology:* defines the interactions between neighboring islands. In static topologies island links remain unchanged throughout the migratory cycle, while in dynamic ones, it is possible to redefine island links during the migratory process, such that island neighborhoods may change.
- **Number of immigrants and emigrants**: gives the number of individuals that are sent and received between two neighboring islands. The related parameter is **NumMigIndividuals** that is the number of individuals for migration.
- *Synchronization*: evolutionary behavior can be synchronous or asynchronous. In synchronous migration, migratory events happen at the same time; in asynchronous migration, each island has its own migratory period, which is the behavior typically found in nature.
- *Migration*: moves or clones a native to a neighboring island. The migrating individuals belong to one of three types: best, worst or random and their fitness (regarding local islands) is used as a classification attribute. If the individual is moved to a neighboring island, there is free space to receive an immigrant on the local island. On the other hand, cloning requires a replacement method to decide which individual will be eliminated on the neighboring island. For doing this, neighboring islands classify individuals in the same types. Three parameters are associated:
  - **EmPolicy** defines the emigration policy: 1) clone native individuals to be sent to the neighboring island; 2) remove native individuals to be sent to the neighboring island;
  - **TypeEmIndividual** defines the type of individuals selected for emigration among: 1) best, 2) worst, and 3) random individuals;
  - **TypeImIndividual** defines the immigration replacement policy that establishes the individuals to be replaced by immigrants on the neighboring island: 1) worst, 2) random, and 3) similar individuals (individuals with the same fitness rank).
- *Migration interval*: number of generations between one migration and another. The associated parameter is **MigrationInterval** that defines the migration interval given from the percentage of breeding cycles (number of generations).

C. Genetic Algorithm

The algorithms are inspired by the Darwinian principle of species evolution [15]. These are probabilistic algorithms that evolve through reproduction with an adaptive search engine based on the survival principle of the fittest. In this work, a simple GA is used with breeding cycle composed by following operators: $selection$, $crossover$, $mutation$ and $replace$. The selection and replacement policy is elitist, selecting the best and replacing the worst individuals in the population. The

mutation process is conservative, a low mutation rate is used, as found in nature, since exaggerated mutations corrupt the genes inherited from parents.

The proposed $\mathcal{GA}$ work as follows. Initially a population of signed permutations is generated from a unsigned permutation input. The breeding cycle is delimited by the input size. Then, for each generation, a set of the best parent individuals in the population is selected, such that each pair of parent individuals gives rise to two new individuals, which undergo mutation, and return to the current population replacing the worst individuals. The pseudo-code of $\mathcal{GA}$ is shown in Algorithm 1.

Algorithm 1: $\mathcal{GA}$ for computing URD

```
Assign values to the parameters;
Generate the initial population;
Compute fitness of the initial population;
for i = 1 to Length(π) do
    Perform selection and save the best solution;
    Apply the crossover operator;
    Apply the mutation operator;
    Compute the fitness of the current population;
    Perform replace operator in the worst individuals;
```

D. Self-adjusting Particle Swarm Optimization

$\mathcal{PSO}$ was introduced in [16] to address continuous domain problems, inspired by the behavior of social organisms in groups, such as species of birds and schools of fish. In $\mathcal{PSO}$ each particle represents a point in the search space and its current position in an n-dimensional space is represented as $x_i = (x_{i1}, \cdots, x_{in})$, j-th dimension. whose motion is specified by the velocity vector $v_i = (v_{i1}, \cdots, v_{in})$ and each particle remembers its best position found during the search as $pbest_i$. In addition, the algorithm maintains the best position with respect to all particles denoted as $gbest$, which is used to guide the particles in the search space. The velocity vector, v_i^{k+1} for the i-particle in the iteration $k+1$ is given by the expression:

$$v_{ij}^{k+1} = w_i^k \cdot v_{ij}^k + c_{1i}^k \cdot rand_{1ij} \cdot (pbest_{ij}^k - x_{ij}^k) + c_{2i}^k \cdot rand_{2ij} \cdot (gbest_j^k - x_{ij}^k) \quad (1)$$

In Eq. (1), w is the weight of inertia (momentum) that introduces friction into the particles motion, reducing inertial velocity; c_1 and c_2 are individual and global acceleration coefficients that influence the maximum step size a particle can take and, $rand_{1i}$ and $rand_{2i}$ are vectors containing random numbers generated at each iteration k, $(0 \leq rand \leq 1)$. The next position of the particle x_i^{k+1} is computed using the following expression: $x_i^{k+1} = x_i^k + v_i^{k+1}$.

To adjust $\mathcal{PSO}$ to URD, each particle x_i in the swarm is associated with a signed permutation constructed from the unsigned genome π provided as input. The component x_{ij}, that is a real value in $(0, 1)$, maps the gene π_j to either $-\pi_j$ or $+\pi_j$, the former case $x_{ij} < 0.5$ and the latter case $x_{ij} \geq 0.5$.

Adaptability in an optimization algorithm can be defined as self-tuning according to rules that have variable parameters. This work uses the self adaptive $\mathcal{PSO}$ proposed in [17], where the parameters w, c_1 and c_2 and the update frequency of each particle are used to perform the self-tuning in the search process (see Algorithm 2). For instance, considering the parameter w, equation $w_i^{k+1} = w_i^k + \alpha_i^k \cdot (w_{\text{best}}^k - w_i^k)$ is applied, where α_i is used to control the diversity of each particle and can take values 0 or *maxIteration*, the maximum number of iterations in $\mathcal{PSO}$. Similarly, for c_1 and c_2

Algorithm 2: Self-adjusting $\mathcal{PSO}$ for computing URD

```
Generate the initial particle swarm;
Initialize Particles using Random Uniform;
for i = 1 to maxIteration do
    Evaluate particles in swarm;
    Update weight w;
    Update gbest;
    Update particles in swarm;
    Update parameters c1 and c2;
```

E. Social Spider Algorithm

Spider locomotion is the subject of bionic engineering research to design robots [18]. One possible reason for this is that most of the observed spiders are solitary [19]. However, some species of spiders are social. For example *Mallos gregalis* live in groups and interact with other spiders in the group. Based on social spiders, a new global optimization method, the Social Spider Algorithm ($\mathcal{SSA}$), was designed in [3].

In $\mathcal{SSA}$, the optimization search space is formulated as a hyper dimensional spider web. A position on the web represents an acceptable solution to the problem and all solutions to the problem belong to that web. The web is also used to transmit spider-generated vibrations. Each spider on the web holds a position whose quality is based on the objective function representing the potential to find a food source in the position. When a spider moves to a new position a vibration is generated which is propagated by the net. A vibration contains information from the propagating spider that will eventually be consulted by other spiders for food source information.

A set of information is required on each spider: 1) position on the web; 2) fitness; 3) vibration of the spider in previous iteration; 4) number of iterations since the spider changed its vibration; 5) movement that the spider performed in the previous iteration; 6) dimension mask represented by a binary vector with size equal to the input that the spider employed to guide movement in the previous iteration. $\mathcal{SSA}$ is described below and its pseudo-code given in Algorithm 3.

a) Initialization: the objective function and the solution space are defined. The value parameters: vibration rate (r_a), probability of changing mask (p_c) and percentage to control updates to mask positions (p_m) are assigned. After setting the values, an initial spider population is created that remains unchanged during $\mathcal{SSA}$ simulation. Spider positions are randomly generated in the search space; the initial target vibration of each spider is set at its current position and the vibration intensity is assigned to be zero.

b) Iteration: in each iteration, all spiders on the web go through the following steps: fitness assessment, vibration generation, mask change, and random walk. The fitness of all spiders is calculated and the overall ideal value updated. Then, spiders generate vibrations in their positions and propagate such vibrations in the web. Each spider will receive vibrations from other spiders according to r_a that consist of the information on source position of the vibration and its attenuation intensity. After receiving the vibrations, the current spider chooses the strongest vibration called v_s^{best} and compares it to its current vibration denoted as v_s^{local}, if v_s^{best} is higher, v_s^{local} updates the value by the value in v_s^{best} and updates the vibration change counter denoted c_s to zero, otherwise it increments c_s by 1 and the value in v_s^{local} is kept. Subsequently, the current spider performs a random walk toward v_s^{local} using the mask dimension to guide the movements. Each spider maintains its mask, where initially all values are zero. In each iteration, the current spider has probability $1 - {p_c}^{c_s}$ to change its mask with $p_c \in (0, 1)$. If the mask changes, each bit of the mask has a p_m probability of being changed.

As applied to $\mathcal{PSO}$, we adapted the case-study to $\mathcal{SSA}$ where a spider's position is associated with a signed permutation constructed from the unsigned permutation π provided as input as following: given a spider s in the position $P_s = (P_{s_1}, \ldots, P_{s_n})$, the value of P_{s_i} belongs to the interval $(0, 1)$, $1 \leq i \leq n$, and a gene π_j is mapped into $-\pi_j$ case $P_{s_i} < 0.5$ or $+\pi_j$ when $P_{s_i} \geq 0.5$. The fitness is given by the reversal distance of the signed permutation generated from the position of the spider s.

Algorithm 3: $\mathcal{SSA}$ for Computing URD

```
Assign values to the parameters;
Generate the initial population Pop;
Initialize v^local for each spider in Pop;
for i = 1 to maxIteration do
    for j = 1 to size(Pop) do
        s ∈ Pop;
        Evaluate the fitness value of s;
        Generate a vibration at the position of s;
    for j = 1 to size(Pop) do
        s ∈ Pop;
        Calculate the intensity of the vibrations V
          generated by all spiders;
        Select the strongest vibration v_s^best from V;
        if Intensity(v_s^best) > Intensity(v_s^local) then
            Store v_s^best as v_s^local;
        Update c_s;
        Generate a random number r between (0, 1);
        if r > 1 − p_c^{c_s} then
            Update the mask dimension using the
              parameter p_m;
        Perform random walk for s;
        Update position for s using the mask
          dimension;
```

III. Related Work

Several papers showed the effectiveness of PIMs. For instance, analyzing the *number of immigrants and emigrants*, [20] worked with the hypothesis that a relatively small migration of individuals is sufficient to disperse genetic material. The effects of *migration* policy in PIMs were investigated in [21]. The authors proposed a model which adjusts automatically the migration interval according to the development of the local population. Also, a migration policy for PIMs with target island defined by attractiveness was proposed in [22]. Subsequently, [23] presented a new evaluation strategy that changes the way to define the attractiveness between islands, in such a manner that islands become more or less attractive according to the quality of their solutions.

Migration topology was the focus in [24], where a master-slave scheme was proposed with slave islands running a genetic algorithm and sending periodically their best partial results to the master island. Also, [25] explored *migration topology*, proposing a new PIM concept with a unidirectional (asynchronous) ring communication topology based on probability models. [13] surveyed the design and analysis of parallel evolutionary algorithms increasing the understanding on how it is possible to obtain accelerations on sequential algorithms.

The influence of *migration interval* together with *number of immigrants and emigrants* for PIMs was studied in [26] observing from the experiments that the *migration interval* seems to be a dominant factor, with *number of immigrants and emigrants* generally being less relevant. *Synchronization policy* in PIMs was studied in terms of the advantages and disadvantages of synchronous and asynchronous migration in [27] concluding that the latter provides better results.

The authors focused in [28] on PIMs with migration topology using a sequential GA to solve the *unsigned translocation distance* problem, an $\mathcal{NP}$-hard problem related to evolutionary distance. The proposed PIMs used ring and complete graph communication topologies with a regular migration policy repeated at each generation, and reused parameters calibrated for the sequential GA. However, accuracy and performance provided by these PIMs were not better than those of the sequential GA. Later, the parameters related to migration policy and breeding cycle have gone through a setup phase to select the best possible values within a range of possible values over the PIMs considering static topologies: torus, complete graph, tree, ring and net, and a migration dynamics which explores the characteristics of individuals [14]. The results obtained were satisfactory, with PIMs presenting good performances and accuracy regarding the sequential GA. For URD, [7] proposed PIMs that use static and dynamic topologies applied in [14]. The proposed PIMs provided better accuracy and performance than the sequential GA.

Important questions about PIMs have arisen, such as: there are parameter settings that increase performance or accuracy?

does the number of islands impacts the final result? are static better than dynamic topologies? and if PIMs submitted to different problems present the same relative behavior? These questions have been explored in [8] using four distant $\mathcal{NP}$-complete problems as case-studies. For each case-study, models using 12 and 24 islands were implemented using static and dynamic topologies. From results, experiments and statistical tests, it was observed that there is no a generic set of parameter values to be fixed into PIMs, in order to achieve the best accuracy and performance. However, overall the best performances were achieved by models with 12 islands, while the best accuracies were obtained by models with 24 islands. Regarding static and dynamic models, in general, dynamic models presented better solutions.

IV. Proposed Parallel Island Models

PIMs have been proposed for which $\mathcal{GA}$, $\mathcal{SSA}$ and $\mathcal{PSO}$ are homogeneously responsible for the evolution of the native individuals on each island. For each algorithm, homogeneous PIM versions using static topologies (tree, net, ring, full graph, torus) and a dynamic topology based on the genotype of the individuals were implemented. To build the dynamic links between the islands a qualification method is used that consists in: initially, computing the fitness average of the islands; and then, measuring the diversity on each island using the variance metric, where high variance means little similarity between native individuals. From the average and variance a key to rank and qualify the islands as good, bad and average, is created. Since the objective of the case-study addressed is minimization, islands with low fitness average and high variance are considered good islands.

Prefixes $\mathcal{G}$, $\mathcal{P}$, $\mathcal{S}$ are used as nomenclature of PIMs from the $\mathcal{GA}$, $\mathcal{PSO}$ and $\mathcal{SSA}$, respectively. Also, the nomenclature uses suffixes identifying the used topology and number of islands.

a) Static Island Models: 30 static PIMs were proposed considering topologies (**TO**) torus, (**F**) full graph, (**N**) net, (**R**) ring and (**TR**) binary tree. Two possible scenarios are observed: topologies **TO** and **F** have dense communication with greater spread of genes through the neighborhood and, **N**, **R** and **TR** a spread of slower genes, giving islands the opportunity to keep their native genes longer. Static PIM topologies were proposed using 12 and 24 islands; for instance, $\mathcal{G}_{\text{TO12}}$, $\mathcal{S}_{\text{F24}}$, $\mathcal{P}_{\text{N12}}$ $\mathcal{P}_{\text{R12}}$ and $\mathcal{P}_{\text{TR24}}$. Net topologies with 12 and 24 processors use 4×3-net and 6×4-net topology, respectively.

b) Dynamics Island Models: 18 dynamic PIMs with 12 and 24 islands are proposed discriminated according to communication between islands with the same classification: good-good, medium-medium, bad-bad (same), and good-bad and medium-medium (gbmm), and random island communication (Rand); for instance, $\mathcal{G}_{\text{same24}}$, $\mathcal{S}_{\text{gbmm24}}$ and $\mathcal{P}_{\text{Rand12}}$.

The extended version of this paper available at http://genoma.cic.unb.br describes the PIMs.

V. Experiments: Performance and Accuracy

The algorithms were developed using the `MPI` library of the `C` language and the experiments executed on a computer with 256GB of RAM, and two processors Xeon E5-2620 with hyper-threading, 6 cores and a CPU clock rate of 2.4Ghz.

TABLE I
Estimated Values for the Migration Parameters

Parameter	estimated values
NumMigIndividuals	1,2,3,4,5,6,7,8,9,10,11,12,13,14
TypeEmIndividual	1=Best, 2=Worst, 3=Random
EmPolicy	1=Clone, 2=Remove
TypeImIndividual	1=Worst, 2=Random, 3=Similar
MigrationInterval	$5\%, 10\%, \cdots, 95\%, 100\%$

The proposed PIMs and their sequential versions have populations of the same size, always $24n$, where n is the length of the input. Thus 12- and 24-island models have respectively island populations with $2n$ and n individuals. The total number of cycles in the evolutionary process is fixed as n.

Estimated values for $\mathcal{GA}$ and $\mathcal{SSA}$ parameters *Crossover, Selection* and *Replacement* and for r_a, p_c and p_m are $2\%, 4\%, \ldots, 98\%, 100\%$ and for $\mathcal{GA}$ parameter *Mutation* $1\%, 2\%, \ldots, 5\%$, and for migration are given in Table I. For each parameter, the models were submitted to evaluation by testing each referenced value. In the end, the parameters that provided the best solutions (Tables II, III, IV) were selected. For the experiments groups of twenty permutations with n genes for $n \in \{50, 60, \ldots, 140, 150\}$ were used.

The accuracy of the proposed PIMs is measured with sets of one hundred unsigned gene permutations used in [6] for each length $n \in \{100, 110, \ldots, 140, 150\}$. For each permutation in the six sets, all the proposed PIMs and the sequential algorithms were executed ten times. Then, the average of the ten results was calculated. These averages represent the number of reversals for each permutation. The average results and standard deviation for the six sets and each algorithm are shown in Tables V,VI,VII, VIII, IX, X. Regarding performance, Table XI shows the speed-up for each PIM taking as input the set of one hundred permutations of length 150.

VI. Discussion

The best overall parameter settings for the proposed PIMs are discussed and then the results obtained are analyzed.

- **NumMigIndividuals** $\mathcal{GA}$: the parameter sets with 5 and 4 individuals were used by 31.25% and 25% of all PIMs, respectively. $\mathcal{SSA}$: 75% of all PIMs use 3 individuals for each migration process. Only one PIM, $\mathcal{S}_{\text{R12}}$, utilizes many individuals (10) in the migration. The remaining PIMs use 2 ($\mathcal{S}_{\text{F12}}$), 5 ($\mathcal{S}_{\text{Rand24}}$) and 7 ($\mathcal{S}_{\text{TO12}}$) individuals. $\mathcal{PSO}$: there is no complete domain, but the most prominent configuration brings 31.25% from all PIMs using 5 individuals.
- **TypeEmIndividual** $\mathcal{GA}$ and $\mathcal{SSA}$: the settings in which the best and worst individuals migrate were dominant with 87.5% and 68.75%, respectively. $\mathcal{PSO}$: the setting in which the best individuals emigrate was dominant with 81.25%, and no PIM uses the worst.
- **EmPolicy** $\mathcal{GA}$: the strategy of removing individuals in emigration is dominant with 81.25% of all PIMs. $\mathcal{SSA}$: the strategy of cloning individuals in emigration was unanimous. $\mathcal{PSO}$: 56.25% of the proposed PIMs use the clone and the remaining the removal strategy.

TABLE II

PARAMETER SETTINGS FOR $\mathcal{GA}$ AND *PIMs* FROM THE $\mathcal{GA}$. 1=$\mathcal{G}_{\text{R12}}$, 2=$\mathcal{G}_{\text{R24}}$, 3= $\mathcal{G}_{\text{F24}}$, 4=$\mathcal{G}_{\text{F12}}$, 5=$\mathcal{G}_{\text{TR12}}$, 6=$\mathcal{G}_{\text{TR24}}$, 7=$\mathcal{G}_{\text{TO12}}$, 8=$\mathcal{G}_{\text{TO24}}$, 9=$\mathcal{G}_{\text{N12}}$, 10=$\mathcal{G}_{\text{N24}}$, 11=$\mathcal{G}_{\text{SAME12}}$, 12=$\mathcal{G}_{\text{SAME24}}$, 13=$\mathcal{G}_{\text{GBMM12}}$, 14=$\mathcal{G}_{\text{GBMM24}}$, 15=$\mathcal{G}_{\text{RAND12}}$, 16= $\mathcal{G}_{\text{RAND24}}$.

Parameter	$\mathcal{GA}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Crossover probability	88%	94%	82%	86%	90%	94%	92%	56%	92%	90%	90%	92%	92%	90%	92%	82%	82%
Mutation probability	1.2%	1%	1%	1%	1%	1%	1%	1%	1.4%	1%	1%	1%	1%	1%	1.4%	2%	1.1%
Pct. for selection	96%	92%	84%	88%	94%	96%	96%	84%	82%	88%	94%	90%	96%	97%	96%	96%	90%
Pct. for replacement	70%	70%	92%	30%	62%	30%	78%	60%	88%	30%	70%	72%	30%	.62%	50%	50%	30%
NumMigIndividuals		12	7	5	11	8	5	4	2	4	5	7	5	13	11	5	13
TypeEmIndividual		1	1	1	1	1	1	3	2	1	1	1	1	1	1	1	1
EmPolicy		2	2	1	1	2	2	2	2	1	2	2	2	2	2	2	2
TypeImIndividual		3	1	1	1	1	1	2	1	1	1	1	1	3	1	1	1
MigrationInterval		30%	85%	30%	25%	30%	30%	95%	90%	10%	20%	10%	10%	30%	10%	10%	25%

TABLE III

PARAMETER SETTINGS FOR $\mathcal{SSA}$ AND *PIMs* FROM THE $\mathcal{SSA}$. 1=$\mathcal{S}_{\text{R12}}$, 2=$\mathcal{S}_{\text{R24}}$, 3= $\mathcal{S}_{\text{F12}}$, 4=$\mathcal{S}_{\text{F24}}$, 5=$\mathcal{S}_{\text{TR12}}$, 6=$\mathcal{S}_{\text{TR24}}$, 7=$\mathcal{S}_{\text{TO12}}$, 8=$\mathcal{S}_{\text{TO24}}$, 9=$\mathcal{S}_{\text{N12}}$, 10=$\mathcal{S}_{\text{N24}}$, 11=$\mathcal{S}_{\text{SAME12}}$, 12=$\mathcal{S}_{\text{SAME24}}$, 13=$\mathcal{S}_{\text{GBMM12}}$, 14=$\mathcal{S}_{\text{GBMM24}}$, 15=$\mathcal{S}_{\text{RAND12}}$, 16= $\mathcal{S}_{\text{RAND24}}$.

Parameter	$\mathcal{SSA}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
r_a	12%	30%	30%	20%	34%	40%	48%	58%	62%	30%	28%	54%	07%	66%	76%	74%	54%
p_c	20%	62%	40%	68%	40%	62%	54%	62%	78%	86%	86%	80%	40%	70%	40%	82%	64%
p_m	75%	50%	96%	88%	66%	30%	84%	50%	86%	94%	96%	94%	64%	76%	96%	76%	84%
NumMigIndividuals		10	3	3	2	3	3	7	3	3	3	3	3	3	3	3	5
TypeEmIndividual		3	3	3	1	1	2	2	2	2	2	2	2	2	2	2	2
EmPolicy		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TypeImIndividual		1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1
MigrationInterval		30%	30%	30%	30%	15%	20%	30%	30%	30%	60%	25%	10%	30%	25%	30%	20%

TABLE IV

PARAMETER SETTINGS FOR THE *PIMs* FROM THE $\mathcal{PSO}$. 1=$\mathcal{P}_{\text{R12}}$, 2=$\mathcal{P}_{\text{R24}}$, 3= $\mathcal{P}_{\text{F12}}$, 4=$\mathcal{P}_{\text{F24}}$, 5=$\mathcal{P}_{\text{TR12}}$, 6=$\mathcal{P}_{\text{TR24}}$, 7=$\mathcal{P}_{\text{TO12}}$, 8=$\mathcal{P}_{\text{TO24}}$, 9=$\mathcal{P}_{\text{N12}}$, 10=$\mathcal{P}_{\text{N24}}$, 11=$\mathcal{P}_{\text{SAME12}}$, 12=$\mathcal{P}_{\text{SAME24}}$, 13=$\mathcal{P}_{\text{GBMM12}}$, 14=$\mathcal{P}_{\text{GBMM24}}$, 15=$\mathcal{P}_{\text{RAND12}}$, 16= $\mathcal{P}_{\text{RAND24}}$.

Parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NumMigIndividuals	12	3	5	6	13	4	9	5	10	5	6	8	5	4	7	2
TypeEmIndividual	3	1	1	1	1	1	1	1	1	1	1	3	3	1	1	1
EmPolicy	2	1	1	1	2	1	2	2	2	2	1	2	1	2	1	2
TypeImIndividual	1	1	1	1	1	1	2	2	1	2	1	2	1	1	1	1
MigrationInterval	20%	10%	25%	30%	15%	10%	30%	25%	30%	20%	10%	10%	10%	10%	30%	10%

TABLE V

INPUT LENGTH (L), AVERAGE RESULTS (A) AND STANDARD DEVIATION (SD) WITH SEQUENTIAL $\mathcal{GA}$ AND STATIC TOPOLOGIES: RING, FULL GRAPH, TREE, TORUS AND NET. 1=$\mathcal{G}_{\text{R12}}$, 2=$\mathcal{G}_{\text{R24}}$, 3= $\mathcal{G}_{\text{F24}}$, 4=$\mathcal{G}_{\text{F12}}$, 5=$\mathcal{G}_{\text{TR12}}$, 6=$\mathcal{G}_{\text{TR24}}$, 7=$\mathcal{G}_{\text{TO12}}$, 8=$\mathcal{G}_{\text{TO24}}$, 9=$\mathcal{G}_{\text{N12}}$, 10=$\mathcal{G}_{\text{N24}}$.

	$\mathcal{GA}$		1		2		3		4		5		6		7		8		9		10	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	89.72	1.04	78.93	1.69	79.67	1.69	79.08	1.75	79.33	1.68	**78.58**	1.73	79.29	1.7	79.03	1.77	79.42	1.7	79.03	1.79	79.19	1.68
110	99.51	0.85	87.81	1.66	88.48	1.64	87.92	1.65	88.21	1.61	**87.38**	1.69	88.17	1.65	87.86	1.6	88.32	1.59	87.82	1.78	88.08	1.63
120	96.36	1.9	96.56	1.66	97.39	1.62	96.7	1.61	97.02	1.63	**96.07**	1.67	97.0	1.64	96.67	1.73	97.05	1.6	96.74	1.73	96.88	1.65
130	105.11	2.14	105.53	1.84	106.36	1.75	105.67	1.79	106.05	1.7	**104.95**	1.7	106.03	1.67	105.5	1.74	106.03	1.68	105.63	1.77	105.82	1.82
140	113.69	2.41	114.16	2.02	115.01	2.02	114.32	2.06	114.7	1.99	**113.57**	2.09	114.62	2.0	114.17	2.05	114.75	2.04	114.23	2.21	114.43	2.07
150	122.7	1.81	123.11	1.53	123.99	1.45	123.32	1.58	123.63	1.49	**122.45**	1.55	123.6	1.49	123.05	1.45	123.57	1.5	123.14	1.57	123.43	1.42

TABLE VI

INPUT LENGTH (L), AVERAGE RESULTS (A) AND STANDARD DEVIATION (SD) FOR THE EXPERIMENT WITH SEQUENTIAL $\mathcal{GA}$ AND DYNAMIC TOPOLOGY. 11=$\mathcal{G}_{\text{SAME12}}$ 12=$\mathcal{G}_{\text{SAME24}}$, 13=$\mathcal{G}_{\text{GBMM12}}$, 14=$\mathcal{G}_{\text{GBMM24}}$, 15=$\mathcal{G}_{\text{RAND12}}$, 16= $\mathcal{G}_{\text{RAND24}}$.

	$\mathcal{GA}$		11		12		13		14		15		16	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	89.72	1.04	78.93	1.77	79.15	1.69	**78.9**	1.64	79.1	1.8	**78.9**	1.79	79.41	1.79
110	99.51	0.85	87.68	1.69	87.96	1.58	87.73	1.67	87.89	1.71	**87.66**	1.73	88.23	1.63
120	**96.36**	1.9	96.49	1.73	96.8	1.68	96.54	1.62	96.68	1.63	96.48	1.7	97.1	1.67
130	**105.11**	2.14	105.34	1.79	105.8	1.77	105.45	1.76	105.63	1.74	105.39	1.84	106.06	1.69
140	**113.69**	2.41	113.96	2.12	114.43	1.98	114.05	2.07	114.21	2.09	114.04	2.12	114.84	2.01
150	**122.7**	1.81	122.97	1.54	123.3	1.49	123.02	1.48	123.26	1.6	123.04	1.51	123.69	1.45

TABLE VII

INPUT LENGTH (L), AVERAGE RESULTS (A) AND STANDARD DEVIATION (SD) WITH SEQUENTIAL $\mathcal{SSA}$ AND STATIC TOPOLOGIES: RING, FULL GRAPH, TREE, TORUS AND NET. 1=$\mathcal{S}_{\text{R12}}$, 2=$\mathcal{S}_{\text{R24}}$, 3= $\mathcal{S}_{\text{F12}}$, 4=$\mathcal{S}_{\text{F24}}$, 5=$\mathcal{S}_{\text{TR12}}$, 6=$\mathcal{S}_{\text{TR24}}$, 7=$\mathcal{S}_{\text{TO12}}$, 8=$\mathcal{S}_{\text{TO24}}$, 9=$\mathcal{S}_{\text{N12}}$, 10=$\mathcal{S}_{\text{N24}}$.

	$\mathcal{SSA}$		1		2		3		4		5		6		7		8		9		10	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	**89.72**	1.04	90.79	0.86	91.11	0.9	90.85	0.88	91.14	0.81	90.76	0.86	91.19	0.83	90.84	0.86	91.18	0.89	90.85	0.81	91.16	0.87
110	**99.51**	0.85	100.72	0.81	101.01	0.76	100.68	0.73	101.03	0.83	100.73	0.8	101.03	0.8	100.7	0.82	101.13	0.78	100.71	0.75	101.05	0.88
120	**109.31**	0.81	110.5	0.77	110.83	0.75	110.54	0.79	110.89	0.8	110.45	0.79	110.88	0.79	110.48	0.76	110.82	0.77	110.5	0.73	110.88	0.75
130	**119.2**	0.91	120.37	0.82	120.72	0.77	120.38	0.79	120.67	0.8	120.3	0.83	120.78	0.79	120.37	0.83	120.71	0.79	120.4	0.85	120.72	0.81
140	**129.0**	0.89	130.19	0.85	130.52	0.83	130.16	0.81	130.48	0.82	130.09	0.89	130.47	0.78	130.12	0.86	130.51	0.84	130.2	0.84	130.5	0.87
150	**138.75**	0.7	139.97	0.67	140.26	0.71	139.94	0.72	140.25	0.72	139.92	0.77	140.33	0.64	139.92	0.74	140.25	0.66	139.95	0.72	140.32	0.72

TABLE VIII

INPUT LENGTH (L), AVERAGE RESULTS (A) AND STANDARD DEVIATION (SD) WITH SEQUENTIAL $\mathcal{SSA}$ AND DYNAMIC TOPOLOGY. 11=$\mathcal{S}_{\text{SAME12}}$ 12=$\mathcal{S}_{\text{SAME24}}$, 13=$\mathcal{S}_{\text{GBMM12}}$, 14=$\mathcal{S}_{\text{GBMM24}}$, 15=$\mathcal{S}_{\text{RAND12}}$, 16= $\mathcal{S}_{\text{RAND24}}$.

	$\mathcal{SSA}$		11		12		13		14		15		16	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	**89.72**	1.04	90.88	0.86	91.25	0.84	90.83	0.87	91.21	0.83	90.91	0.85	91.17	0.86
110	**99.51**	0.85	100.69	0.78	101.04	0.78	100.67	0.8	101.0	0.83	100.72	0.73	101.06	0.75
120	**109.31**	0.81	110.48	0.8	110.89	0.74	110.52	0.84	110.86	0.76	110.54	0.79	110.88	0.74
130	**119.2**	0.91	120.42	0.78	120.74	0.81	120.45	0.8	120.76	0.81	120.36	0.83	120.73	0.82
140	**129.0**	0.89	130.17	0.89	130.48	0.9	130.16	0.83	130.52	0.83	130.19	0.84	130.55	0.8
150	**138.75**	0.7	139.94	0.7	140.3	0.71	139.92	0.71	140.28	0.71	139.96	0.73	140.28	0.72

TABLE IX

INPUT LENGTH (L), AVERAGE RESULTS (A) AND STANDARD DEVIATION (SD) WITH SEQUENTIAL $\mathcal{PSO}$ AND STATIC TOPOLOGIES: RING, FULL GRAPH, TREE, TORUS AND NET. 1=$\mathcal{P}_{\text{R12}}$, 2=$\mathcal{P}_{\text{R24}}$, 3= $\mathcal{P}_{\text{F12}}$, 4=$\mathcal{P}_{\text{F24}}$, 5=$\mathcal{P}_{\text{TR12}}$, 6=$\mathcal{P}_{\text{TR24}}$, 7=$\mathcal{P}_{\text{TO12}}$, 8=$\mathcal{P}_{\text{TO24}}$, 9=$\mathcal{P}_{\text{N12}}$, 10=$\mathcal{P}_{\text{N24}}$.

	$\mathcal{PSO}$		1		2		3		4		5		6		7		8		9		10	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	83.11	1.49	81.68	1.59	81.49	1.51	81.97	1.5	81.99	1.48	**81.4**	1.53	81.43	1.55	81.83	1.64	**81.4**	1.63	81.66	1.62	**81.4**	1.63
110	92.37	1.44	90.88	1.36	90.82	1.37	91.27	1.33	91.24	1.31	**90.6**	1.4	90.70	1.52	90.9	1.45	**90.6**	1.43	90.8	1.38	**90.6**	1.43
120	101.71	1.52	100.18	1.49	100.04	1.49	100.47	1.41	100.58	1.34	**99.68**	1.44	99.87	1.51	100.05	1.55	99.86	1.5	100.08	1.45	99.86	1.5
130	111.02	1.53	109.58	1.47	109.32	1.49	109.84	1.28	109.94	1.45	**109.04**	1.53	109.17	1.48	109.34	1.51	109.17	1.51	109.39	1.45	109.17	1.51
140	120.3	1.7	118.6	1.7	118.3	1.7	118.97	1.59	119.05	1.56	**118.02**	1.68	118.16	1.83	118.39	1.75	118.11	1.84	118.49	1.73	118.09	1.84
150	129.41	1.26	127.87	1.34	127.69	1.29	128.35	1.27	128.32	1.28	**127.36**	1.24	127.45	1.37	127.71	1.58	127.48	1.38	127.75	1.33	127.48	1.38

TABLE X

INPUT LENGTH (L), AVERAGE RESULTS (A) AND STANDARD DEVIATION (SD) WITH SEQUENTIAL $\mathcal{PSO}$ AND DYNAMIC TOPOLOGY. 11=$\mathcal{P}_{\text{SAME12}}$, 12=$\mathcal{P}_{\text{SAME24}}$, 13=$\mathcal{P}_{\text{GBMM12}}$, 14=$\mathcal{P}_{\text{GBMM24}}$, 15=$\mathcal{P}_{\text{RAND12}}$, 16= $\mathcal{P}_{\text{RAND24}}$.

	$\mathcal{PSO}$		11		12		13		14		15		16	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	83.11	1.49	81.71	1.58	81.66	1.45	81.57	1.61	81.57	1.62	81.74	1.42	**81.4**	1.62
110	92.37	1.44	90.95	1.43	90.93	1.47	90.81	1.42	90.8	1.38	90.99	1.36	**90.71**	1.53
120	101.71	1.52	100.22	1.47	100.14	1.43	100.05	1.47	100.08	1.49	100.22	1.4	**99.98**	1.46
130	111.02	1.53	109.44	1.47	109.53	1.47	109.25	1.42	109.32	1.56	109.56	1.39	**109.18**	1.54
140	120.3	1.7	118.56	1.75	118.5	1.78	118.37	1.8	118.5	1.76	118.73	1.68	**118.20**	1.82
150	129.41	1.26	127.88	1.29	128.03	1.24	127.66	1.28	127.68	1.33	128.06	1.22	**127.61**	1.39

TABLE XI

SPEED-UP COMPUTED FOR EACH PROPOSED PIM.

$\mathcal{G}_{\text{R12}}$	$\mathcal{G}_{\text{R24}}$	$\mathcal{G}_{\text{F12}}$	$\mathcal{G}_{\text{F24}}$	$\mathcal{G}_{\text{TR12}}$	$\mathcal{G}_{\text{TR24}}$	$\mathcal{G}_{\text{TO12}}$	$\mathcal{G}_{\text{TO24}}$	$\mathcal{G}_{\text{N12}}$	$\mathcal{G}_{\text{N24}}$	$\mathcal{G}_{\text{same12}}$	$\mathcal{G}_{\text{same24}}$	$\mathcal{G}_{\text{gbmm12}}$	$\mathcal{G}_{\text{gbmm24}}$	$\mathcal{G}_{\text{Rand12}}$	$\mathcal{G}_{\text{Rand24}}$
6.11	**7.96**	7.51	7.62	6.47	7.22	7.38	7.28	7.24	7.26	7.27	7.83	6.74	7.20	6.42	7.85
$\mathcal{S}_{\text{R12}}$	$\mathcal{S}_{\text{R24}}$	$\mathcal{S}_{\text{F12}}$	$\mathcal{S}_{\text{F24}}$	$\mathcal{S}_{\text{TR12}}$	$\mathcal{S}_{\text{TR24}}$	$\mathcal{S}_{\text{TO12}}$	$\mathcal{S}_{\text{TO24}}$	$\mathcal{S}_{\text{N12}}$	$\mathcal{S}_{\text{N24}}$	$\mathcal{S}_{\text{same12}}$	$\mathcal{S}_{\text{same24}}$	$\mathcal{S}_{\text{gbmm12}}$	$\mathcal{S}_{\text{gbmm24}}$	$\mathcal{S}_{\text{Rand12}}$	$\mathcal{S}_{\text{Rand24}}$
82.39	114.12	79.04	**116.71**	82.71	114.35	78.97	110.66	81.64	116.02	81.66	110.18	78.77	106.79	78.44	113.84
$\mathcal{P}_{\text{R12}}$	$\mathcal{P}_{\text{R24}}$	$\mathcal{P}_{\text{F12}}$	$\mathcal{P}_{\text{F24}}$	$\mathcal{P}_{\text{TR12}}$	$\mathcal{P}_{\text{TR24}}$	$\mathcal{P}_{\text{TO12}}$	$\mathcal{P}_{\text{TO24}}$	$\mathcal{P}_{\text{N12}}$	$\mathcal{P}_{\text{N24}}$	$\mathcal{P}_{\text{same12}}$	$\mathcal{P}_{\text{same24}}$	$\mathcal{P}_{\text{gbmm12}}$	$\mathcal{P}_{\text{gbmm24}}$	$\mathcal{P}_{\text{Rand12}}$	$\mathcal{P}_{\text{Rand24}}$
11.14	9.49	11.16	9.32	**11.39**	9.30	10.93	9.41	11.02	9.21	10.97	9.28	10.92	9.31	10.81	9.40

- **TypeImIndividual** $\mathcal{GA}$: removing the worst individuals on target island was the strategy in 81.25% of PIMs. $\mathcal{SSA}$: 87.25% of all PIMs remove the worst individuals on target island and the remaining remove similar individuals. $\mathcal{PSO}$: 75% of all PIMs remove the worst individuals on target island, and the remaining remove random individuals.
- **MigrationInterval** $\mathcal{GA}$: the migration interval with 10% and 30% of the breeding cycle was the most popular configuration used by 62.50% of all PIMs. $\mathcal{SSA}$: 56.25% of the PIMs use the configuration with 30% of maximum generation. $\mathcal{S}_{\text{N24}}$ is the only PIM that starts the migratory process late, only after reaching 60% of generation. $\mathcal{PSO}$: 10% and 30% settings of the maximum generation were used by 43.75% and 25% of the PIMs, respectively.

Since URD is a minimization problem, the smaller the output the higher the accuracy. Analyzing the accuracy of the results, the PIMs proposed from the $\mathcal{SSA}$ provided lower quality solutions than $\mathcal{SSA}$. However, the quality of these solutions are inferior to those given by $\mathcal{PSO}$ and $\mathcal{GA}$ and their respective parallel versions.

From the results of $\mathcal{PSO}$ and its parallel versions, we have the following scenarios: all proposed PIMs provided better solutions than $\mathcal{PSO}$. Comparing the static and dynamic models, as shown in Tables IX, X, the best solutions were provided by the static PIM ($\mathcal{P}_{\text{TR12}}$). In addition, analyzing the best dynamic PIM ($\mathcal{P}_{\text{Rand24}}$), there are 4 static PIMs ($\mathcal{P}_{\text{TR12}}$, $\mathcal{P}_{\text{TR24}}$, $\mathcal{P}_{\text{TO24}}$, $\mathcal{P}_{\text{N24}}$) with superior solutions that $\mathcal{P}_{\text{Rand24}}$.

For PIMs proposed from $\mathcal{GA}$, the best solutions were computed by $\mathcal{G}_{\text{TR12}}$ that provided the best solutions for the case-study in general, surpassing all other algorithms.

An interesting observation is the impact caused by the variation in the size of island populations. For this work, islands with n individuals were chosen, motivated by the time taken by $\mathcal{SSA}$ when running with $n \log n$ spiders. Looking at Table VI, and comparing with results in [14], [7], [6], where PIMs from $\mathcal{GA}$ with island populations of size $n \log n$ presented better results than $\mathcal{GA}$, it is observed that decreasing the island populations was not beneficial since here some PIMs are overcome by their sequential version.

The best speed-ups were provided by PIMs from $\mathcal{SSA}$, however, it does not generate any enthusiasm to improve speed-ups, since solutions are much worse than the provided

by $\mathcal{SSA}$. For PIMs from $\mathcal{GA}$, speed-ups are smaller than for PIMs from $\mathcal{SSA}$. The best performance is provided by $\mathcal{G}_{\text{R24}}$, but the solutions are no better than those given by $\mathcal{G}_{\text{TR12}}$, which presented the best solutions with speed-up of 6.47. On the other hand, the PIM $\mathcal{P}_{\text{TR12}}$ from $\mathcal{PSO}$ that provided the best results also presented the best speed-up (11.39) regarding PIMs from $\mathcal{PSO}$. In works [14], [7], [6] PIMs with a smaller number of islands had better performances than those with a larger number of islands, but in this work, the best performances analyzing PIMs from $\mathcal{GA}$ and $\mathcal{SSA}$ were given using 24 islands and for PIMs from $\mathcal{PSO}$ using 12 islands.

A statistical analysis was performed over the results of experiments for all PIMs. First the Friedman test was performed an then the Holm test (Post-hoc test). This procedure is taken from [29] (see also [30] and [31]), and was extensively applied in previous related works. The extended version of the paper at http://genoma.cic.unb.br contains the statistical analysis.

VII. Conclusions and future work

Previous works (e.g., [28], [6], [7], [14]) showed that dynamic PIMs offer better solutions than static ones, and better speedups using smaller number of islands. In this work, PIMs were proposed using static and dynamic topologies to analyze island migration. A sample of the problem search space was used on each island, either n or $2n$, where $24n$ is the total population size. With this change previous observations were refuted, since dynamic PIMs presented much lower accuracy than static PIMs and all proposed PIMs from $\mathcal{GA}$ and $\mathcal{SSA}$ with 24 islands presented better speedups than with 12, but vice-versa for PIMs from $\mathcal{PSO}$. From these facts, on concludes that, in addition to an adequate setting of the parameters involved in the migratory process, the number of natives on the islands need to be taken into account to improve the gains in performance and accuracy.

As future work, we intend to continue studying variations in the size of island population and propose heterogeneous PIMs, in which different algorithms are performed on each island. Initially, using $\mathcal{PSO}$, $\mathcal{SSA}$, $\mathcal{GA}$ and later on a variety of other bioinspired algorithms. Such research will require the initial development of specialized sequential versions for these algorithms and the subsequently development of heterogeneous PIMs for which migration policies, and then performance and accuracy will be investigated. Besides, of course, making a comparison between heterogeneous and homogeneous PIMs.

References

[1] C. B. Pettey, M. R. Leuze, and J. J. Grefenstette, "Parallel genetic algorithm," in *2nd Int. Conf. on Genetic Algorithms their applications at the Massachusetts Institute of Technology*, 1987.

[2] N. Eldredge and G. J. Stephan, "Punctuated equilibrium prevails," *Nature*, vol. 332, no. 6161, pp. 211–212, 1988.

[3] J. J. Yu and V. O. Li, "A social spider algorithm for global optimization," *Applied Soft Computing*, vol. 30, pp. 614 – 627, 2015.

[4] A. Caprara, "Sorting by reversals is difficult," in *1st Annual Int. Conf. on Comp. molecular Biology*. ACM, 1997, pp. 75–83.

[5] 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.

[6] 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 *IEEE CEC*, 2017, pp. 741–748.

[7] L. A. da Silveira, J. L. Soncco-Alvarez, T. A. de Lima, and M. Ayala-Rincon, "Parallel multi-island genetic algorithm for sorting unsigned genomes by reversals," in *IEEE CEC*, 2018, pp. 1–8.

[8] L. A. da Silveira, J. L. Soncco-Álvarez, J. B. de Barros, C. H. Llanos, and M. Ayala-Rincón, "On the Behavior of Parallel Island Models," Univ. de Brasília, Tech. Rep., 2019, av. at http://genoma.cic.unb.br.

[9] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for the inversion distance between two chromosomes," in *Comb. Pattern Matching (CPM)*, ser. LNCS, vol. 684. Springer, 1993, pp. 87–105.

[10] 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.

[11] J. L. Soncco-Álvarez and M. Ayala-Rincón, "Sorting permutations by reversals through a hybrid genetic algorithm based on breakpoint elimination and exact solutions for signed permutations," *Electronic Notes in Theoretical Computer Science*, vol. 292, pp. 119–133, 2013.

[12] J. L. Soncco-Álvarez, D. M. Muñoz, and M. Ayala-Rincón, "Opposition-based memetic algorithm and hybrid approach for sorting permutations by reversals," *Evol. Comput.*, vol. 27, no. 2, pp. 229–265, 2019.

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

[14] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón, "Parallel Island Model Genetic Algorithms applied in NP-Hard problems," in *IEEE CEC*, 2019, pp. 1–8.

[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman, 1989.

[16] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *IEEE Int. Conf. on neural networks*, vol. 4, 1995, pp. 1942–1948.

[17] K. Yasuda and K. Yazawa, "Parameter self-adjusting strategy for particle swarm optimization," in *11th Int. Conf. on Intelligent Systems Design and Applications*. IEEE, 2011, pp. 265–270.

[18] Y. Mark, Z. Ying, and D. David, "Modular robots," *IEEE Spectrum*, vol. 39, no. 2, pp. 30–34, 2002.

[19] R. Foelix, *Biology of Spiders*, 3rd ed. Oxford University Press, 2010.

[20] T. C. Belding, "The distributed genetic algorithm revisited," in *6th Int. Conf. on Genetic Algorithms*. Morgan Kaufmann, 1995, pp. 114–121.

[21] T. Hong, W. Lin, S. Liu, and J. Lin, "Experimental analysis of dynamic migration intervals on 0/1 knapsack problems," in *IEEE CEC*, 9 2007, pp. 1163–1167.

[22] G. Duarte, A. Lemonge, and L. Goliatt, "A dynamic migration policy to the island model," in *IEEE CEC*, 2017, pp. 1135–1142.

[23] G. Duarte, A. Lemonge, and L. Goliatt, "A new strategy to evaluate the attractiveness in a dynamic island model," in *IEEE CEC*, 2018, pp. 1–8.

[24] F. J. Marin, O. Trelles-Salazar, and F. Sandoval, "Genetic Algorithms on LAN-message passing architectures using PVM: Application to the Routing problem," in *Int. Conf. Proc. Parallel Problem Solving from Nature (PPSN)*, ser. LNCS, vol. 866. Springer, 1994, pp. 534–543.

[25] J. Jaros and J. Schwarz, "Parallel BMDA with probability model migration," in *IEEE CEC*, 2007, pp. 1059–1066.

[26] Z. Skolicki and K. De Jong, "The influence of migration sizes and intervals on island models," in *7th Annual Conf. on Genetic and Evol. Comp. (GECCO)*, 2005, pp. 1295–1302.

[27] F. Fernández, G. Galeano, and J. A. Gómez, "Comparing Synchronous and Asynchronous Parallel and Distributed Genetic Programming Models," in *5th European Conf. on Genetic Programming*, ser. LNCS, vol. 2278. Springer, 2002, pp. 326–335.

[28] L. A. da Silveira, J. L. Soncco-Álvarez, and M. Ayala-Rincón, "Parallel memetic genetic algorithms for sorting unsigned genomes by translocations," in *IEEE CEC*, 2016, pp. 185–192.

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

[30] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *J. of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[31] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Comp.*, vol. 1, no. 1, pp. 3–18, 2011.