Reconfigurable Heterogeneous Parallel Island Models

Lucas A. da Silveira†, Thaynara A. de Lima‡ and Mauricio Ayala-Rincón†∗

†Departments of Computer Science and ∗Mathematics, Universidade de Brasília, 70900–010 Brasília D.F., Brazil
‡Institute of Mathematics and Statistics, Universidade Federal de Goiás — Campus II, 74690–900 Goiânia, Brazil

Abstract—Heterogeneous Parallel Island Models (HePIMs) run different bio-inspired algorithms (BAs) in their islands. From a variety of communication topologies and migration policies fine-tuned for homogeneous PIMs (HoPIMs), which run the same BA in all their islands, previous work introduced HePIMs that provided competitive quality solutions regarding the best-adapted BA in HoPIMs. This work goes a step forward, maintaining the population diversity provided by HePIMs, and increasing their flexibility, allowing BA reconfiguration on islands during execution: according to their performance, islands may substitute their BAs dynamically during the evolutionary process. Experiments with the introduced architectures were applied to the NP-hard problem of sorting permutations by reversals, using four different BAs, namely, simple Genetic Algorithm, Double-point crossover Genetic Algorithm, Differential Evolution, and self-adjusting Particle Swarm Optimization. The results showed that the novel reconfigurable heterogeneous models compute better quality solutions than the HePIMs closing the gap with the HoPIM running the best-adapted BA.

I. INTRODUCTION

Evolutionary computing relies on the evolution of candidate solutions over a finite number of generations to obtain accurate solutions for complex optimization problems. Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO), among other Bio-inspired algorithms (BAs), have been applied to successfully solve a diversity of such problems applied in engineering and sciences. BAs guide the evolution of a population of individuals (candidate solutions) to improve their fitness to achieve a feasible solution to the target problem. BAs apply specialized computational rules to promote individual information exchange to the benefit of the population. However, optimization through such BA approaches demands exhaustive computations and efficient resources. Parallelization is one of the natural mechanisms applied to speed up and improve the accuracy of solutions obtained by BAs.

This work studies Parallel Island Models (PIMs) that partition the population among their islands (processors) and simultaneously run a BA in each island. In addition to the exchange of individuals, PIMs promote migration between islands. When all islands run the same BA, such models are called homogeneous PIMs (HoPIMs). This work improves heterogeneous PIMs (HePIMs) [1], in which islands may run different BAs, allowing algorithmic reconfiguration on their islands, i.e., islands may dynamically update their BAs (maintaining their populations). In addition to an adequate and well-calibrated migration policy required by HePIMs, reconfigurable HePIMs exchange information between islands to decide how islands should reconfigure their BAs.

Silveira et al. [1] introduced HePIMs running four different BAs in their islands, namely, single-point crossover elitist Genetic Algorithm (GA), double-point crossover popular GA (GAD), Differential Evolution (DE), and self-adjusting Particle Swarm Optimization (PSO) (see e.g., [2], [3], and [4]). Choosing two GAs (GA and GAD) is justified because different crossover policies give rise to substantially different performances according to the combinatorics of the target problem (e.g., [5], [6]). Moreover, the aim of empowering PIMs with the novel reconfigurability feature is to enable updating the BAs dynamically in each processor to improve performance. PIMs’ performance depends on a good calibration of the breeding cycle parameters (related to the involved BA) and on vital aspects of the architecture such as island communication synchronism, island migration frequency, communication topology, and migration policy. We select the two most successful asynchronous HePIMs from [1] maintaining their parameters and adding the reconfiguration frequency. The selected PIMs have a tree and a complete-graph topology, respectively, with static and dynamic migration policies.

The new reconfigurable HePIMs are tested in solving the unsigned reversal distance problem (URD), an NP-hard problem ([7]). Approaches to solve URD are applied in combinatorics to explain algebraic aspects of permutations ([8]) and, in genomics, to analyze the evolutionary distance between genomes ([9]), also represented as permutations.

Main contributions. We design new reconfigurable HePIMs that run GA, GAD, DE, and PSO in their islands using two successful asynchronous topologies from [1]. Non-reconfigurable HePIMs computed competitive solutions regarding most HoPIMs. The novel reconfigurable architectures give solutions that improve the quality of those calculated by HePIMs and are very close to the answers provided by best-adapted HoPIMs (running DE in all their islands). The heterogeneity of the new models effectively spreads, through the migration policy, the good results experimented by individuals guided by different BAs in each island to the whole architecture. Furthermore, the novel reconfiguration ability shares the experiences of the BAs on each island. Since island BAs may be updated, the new models with the reconfiguration capability exceed the flexibility of HoPIMs and non-reconfigurable HePIMs.

Organization. Sec. II discusses PIMs, the unsigned reversal distance problem, and the four selected BAs: GA, GAD, DE, and PSO. Sec. III presents related work, and Sec. IV introduces...
the new reconfigurable HePIMs explaining how different BAs are reconfigured. Then, Sec. V presents experiments, discusses accuracy results, speedup, and statistical analysis. Finally, Sec. VI concludes and discusses future work. To assure reproducibility, source and data used in the experiments are available at http://genoma.cic.unb.br.

II. BACKGROUND

A. Parallel island model (PIM)

PIMs were proposed initially for GAs [10] and, besides improving speedup, it is expected that such models also boost the solutions provided by sequential GA.

The population is distributed into islands whose number is determined by the developer, running their BAs in parallel. The connection between the islands establishes the model’s topology. Static PIMs maintain the connections fixed during the execution, whereas dynamic models admit changes during the process. Linked islands exchange individuals to evolve. Such a transfer can be uni- or bi-directionally. Different topologies and strategies to implement them are available (e.g. [11], [12], [13]). Homogeneous PIMs execute the same BA in all islands, whereas heterogeneous models admit different BAs running in their islands.

A migration policy guides the exchange of individuals between islands during the evolutionary process. PIMs have breeding-cycle and migration parameters tuned to improve the quality of solutions. In the following, the migration parameters are briefly presented. Some of them consider the classification of individuals as best, worst and random, based on a rank established according to their fitness. The first half corresponds to the best and the second to the worst individuals, whereas random individuals are selected randomly.

- Individuals number (IN): number of individuals emigrating from each island.
- Emigrant Individuals (EMI): rule the type of individuals selected for emigration among: 1. best, 2. worst, and 3. random.
- Immigrant Individuals (IMI): determines the type of individuals in the target island replaced by immigrants among: 1. worst, 2. random, and 3. similar.
- Emigration Policy (EP): defines whether individuals are cloned or removed in the local island when they emigrate to the target island.
- Migration Interval (MI): is the percentage of iterations (generations) of the evolutionary process after which the migration process is redone. Each island separately evolves its population by MI × maxIt generations, where maxIt is the total number of iterations performed by each BA.

PIMs are classified according to the synchronicity in which islands evolve their population. In synchronous PIMs, islands evolve performing each generation simultaneously, whereas, in asynchronous PIMs, islands evolve independently even during migration. The latest mimics the behavior found in nature.

Here, we introduce reconfigurable heterogeneous PIMs. First, at a fixed generation percentage, called Reconfiguration Frequency (RF), it is checked what islands have the best and the worst solutions regarding a metric based on the fitness average and the variance of the population. Then, the worst island updates its BA to the BA applied by the best island.

B. Case-study

The evolutionary distance between two organisms can be computed as the number of rearrangements needed to transform a genome into another using some evolutionary measure. This work considers the minimum number of reversals to compute the distance between unichromosomal organisms.

Permutations on \(\{1, \cdots, n\}\) represent a genome containing \(n\) genes. Given a genome \(\pi = (\pi_1, \pi_2, \cdots, \pi_n)\), where \(1 \leq i, \pi_i \leq n\), a reversal \(p^k\), for \(1 \leq j \leq k \leq n\), transforms \(\pi\) into \(\pi' = (\cdots, \pi_{j-1}, \pi_k, \cdots, \pi_j, \pi_{k+1}, \cdots)\), that is, it inverts the elements between \(\pi_j\) and \(\pi_k\). Genes may have forward or reverse orientation on a genome; if known, genes receive a positive or negative sign, respectively, and the genome is a signed permutation. There are two evolutionary problems related to computing the distance by reversals. The signed reversal distance (SRD) problem asks for the minimum number of reversals needed to transform a signed permutation into another. Moreover, the unsigned reversal distance (URD) problem, addressed in this paper, consists of computing such a number between unsigned permutations in which the orientation of genes is unknown. It is well-known that SRD belongs to class \(\mathcal{P}\) [14], whereas URD is an \(\mathcal{NP}\)-hard problem [7].

The fitness used by the algorithms is computed over signed permutations, generated after a random assignment of signs to the genes of a permutation.

C. Local Evolutionary Engines — bio-inspired Algorithms

Four BAs, widely used for analyzing optimization problems and having distinct adaptability characteristics are applied.

- Simple Genetic Algorithm (GA): developed by J. H. Holland [2]. To evolve local population. GA considers a breeding cycle where the parents are selected and produce offspring by applying one-point crossover (Fig. 1 (a)). Then, the descendants replace the worst individuals in the current population. The breeding cycle relies on four parameters, namely, the percentages of selection and replacement, and the probability of application of mutation and crossover.
- Double-point Crossover Genetic Algorithm (GAD): it has a similar behavior than GA except by the technique to promote crossover, illustrated in Fig. 1 (b), and how the local population evolves: in contrast with GA descendants replace individuals randomly selected in GAD.
- Differential Evolution (DE): proposed by Storn and Price [3]. It is a method to optimize functions over the real multidimensional space \(\mathbb{R}^n\). We adapt the algorithm by restricting the domain of the function as the set of permutations. The evolutionary process is guided by the mutation factor \(F_M\), applied to individuals randomly selected from the population to generate mutants, and the probability of
crossover $P_C$. The local population evolves by replacing individuals having the worst fitness with mutants.

- Self-adjusting Particle Swarm Optimization (PSO): introduced by Eberhart and Kennedy [4]. It is based on the behavior of social organisms in groups. PSO aims to optimize continuous functions from particles’ velocity and position in an $n$-dimensional space. At each iteration, the particle velocity is updated from the best position of the particle and of all particles, the individual and global acceleration (that influence the distance a particle can cover), and the weight of inertia (momentum). In this paper, we use the PSO proposed in [15], which is self-adaptive since momentum and the individual and global acceleration coefficients are self-tuning during the search process.

To adapt PSO and DE to URD, to each $n$-dimensional real vector $v$ randomly generated is associated a signed permutation constructed from the unsigned permutation $\pi = (\pi_1, \ldots, \pi_n)$ gave as input: if the $i$-th entry of $v$ belongs to the interval $[0, 0.5)$ then $\pi_i$ receives a negative orientation; case such a coordinate is in $[0.5, 1]$ then $\pi$ is assigned positively. However, if the continuous vector representation has an entry outside of the interval $[0, 1]$, a correct orientation is randomly generated. For GA and GAD, the orientation of the genes in each individual is randomly generated as $\pm 1$. After the transformation of an unsigned to signed permutations, the linear algorithm to solve the SRD problem, proposed by Bader et al. [16], computes the fitness of each particle/individual.

$$\begin{align*}
\text{(a) One-point crossover} & \quad 1 \overset{2}{\leftarrow} 3 \overset{4}{\leftarrow} 5 \overset{6}{\leftarrow} 7 \overset{8}{\leftarrow} 9 \Rightarrow 1 \overset{2}{\leftarrow} 3 \overset{4}{\leftarrow} 5 \overset{6}{\leftarrow} 7 \overset{8}{\leftarrow} 9 \\
\text{(b) Double-point crossover} & \quad 1 \overset{2}{\leftarrow} 3 \overset{4}{\leftarrow} 5 \overset{6}{\leftarrow} 7 \overset{8}{\leftarrow} 9 \Rightarrow 1 \overset{2}{\leftarrow} 3 \overset{4}{\leftarrow} 5 \overset{6}{\leftarrow} 7 \overset{8}{\leftarrow} 9
\end{align*}$$

Fig. 1. One-point and double-point crossing operators.

III. RELATED WORK

As far as we know, no HePIM has been proposed that may dynamically update their BAs as proposed in this work. So, we discuss a few works related to non-reconfigurable HePIMs.

Bianchini and Brown [17] proposed HePIMs with ring and torus topologies and applied them to the task map scheduling problem showing that HePIMs compute better solutions than HoPIMs. In addition, they observed that adding islands is better than increasing the population. Also, Lin et al. [18] proposed HePIMs, considering several migration strategies and topologies addressing the graph partitioning problem. They showed that 25-island PIMs are better than the sequential GA, using a migration strategy that replaces the worst individuals on target islands. Furthermore, they showed that exchanging individuals regarding their fitness-based population similarity instead gets good results without speed degradation.

Izzo et al. [19] proposed an asynchronous-migration HePIM from variations of the DE algorithm. Asynchrony was shown to be more intuitive and suitable over TCP/IP, where resources might become available/unavailable at any time. Izzo et al. models performed better than their sequential versions.

Gong and Fukunaga [20] proposed a GA-based HePIM that randomly selects different parameters for each processor; some processors are expected to be assigned parameters performing well on a given problem. Therefore, it may be conceived as a single-cycle reconfigurable model that applies only an initial adjustment of the same algorithm and does not reconfigure BAs dynamically as our architectures do.

Duarte et al. [21] proposed an attractiveness-based migration policy for five-island HePIMs that is based on island solutions’ quality. Attractiveness was adjusted in [11], inspired by the natural phenomenon known as stigmergy [22], and the mechanism to compute islands’ connections.

Silveira et al. [23] proposed HoPIMs for a sequential GA introduced in [24] to solve the unsigned translocation problem. Such PIMs outperformed the accuracy obtained by the GA after carefully calibrating the migration and breeding cycle parameters and exploring a variety of topologies [25], [26]. Further, Silveira et al. [12] analyzed synchronous HoPIMs for GA, PSO, and the Social Spider Algorithm (SSA). Experiments showed that HoPIMs applying PSO and GA are competitive, while those running SSA gave the best speedups but computed the worst-accuracy solutions. Finally, Silveira et al. [1] proposed a variety of HePIMs to deal with URD. In this work, we select Lucas et al. models $P_{best}$ and $P_{dominant}$ for our comparison with the novel reconfigurable HePIMs.

HePIMs also have been conceived to solve multiobjective optimization problems (MOPs). We believe that MOPs are exciting applications to our reconfigurable HePIMs since each island may update its BA to optimize a single objective function. For example, Zang et al. [27] proposed a multi-swarm optimizer that handles each objective function of a MOP with a different slave swarm and a master swarm covers gaps among non-dominated optima using a multiobjective PSO. Also, Xu et al. [28] proposed a model with EAs using two subpopulations to solve dynamic interval MOPs, which are MOPs that change interval parameters of their objectives or constraints over time. In addition, Gong et al. [29] proposed a model that handles a cooperative co-evolutionary MOP based on dynamic interval similarity. Gong et al. approach split decision variables according to their interval similarity and interval parameters. Then, decision variables are optimized cooperatively. Furthermore, Hashimoto et al. [30] proposed a HePIM to solve multi-task problems, where each island evaluates an objective. Migrants are selected at high migration frequency and removed randomly on each local island, replacing the worst individuals in the target islands. Since immigrants went to islands responsible for different objectives, their fitness values are the worst, assuming they have fewer chances of being suitable for the target island objective.

IV. COMMUNICATION TOPOLOGIES

We select a static and a dynamic topology that successfully addressed URD in [12] for homogeneous PIMs and in [1]...
for non reconfigurable heterogeneous PIMs. We choose asynchronous models since the dynamic asynchronous HePIM was the one that provided the best results.

The static topology is a 12-island bi-directional binary tree (tree to the left in Fig. 2), and the dynamic topology is the 12-island complete graph (graph to the left in Fig. 3). In the complete graph topology all pairs of islands may exchange individuals. The island communication dynamism is acquired by exploring diversity and quality into each island, given by fitness variance and average metrics. Variance measures islands’ diversity: high variance represents high individuals’ diversity, improving the chances of evolution into islands. Fitness average measures the quality of island populations. According to such metrics, the islands are ranked as good, bad, and medium. Migrations exchange individuals between good and bad islands, and medium and medium islands only (for short, "gbmm"). Reconfiguration uses the same metric to update the BA executed by islands, according to the best performance experienced by other islands. A master island is responsible for receiving data from all islands; it ranks and notifies the worst island. The worst island replaces its algorithm by the algorithm of the best ranked island.

So, reconfigurable HePIMs perform migration and updating of BAs at some intervals during their evolution.

This paper introduces the models \( P^{gbmm}_{Tr12A} \) and \( P^{gbmm}_{Tr12A} \). The former uses the static tree topology, and the latter one the dynamic complete graph topology. Both topologies are asynchronous and evolve through a refined migration policy that allows the exchange of individuals, maintaining diversity. Furthermore, through the new feature of dynamic reconfiguration that allows updating the BAs executed in their islands, both the population diversity and performance of the island model regarding HePIMs are improved. Reconfiguration cycles for \( P^{gbmm}_{Tr12A} \) and \( P^{gbmm}_{Tr12A} \) are illustrated in Figs. 2 and 3, respectively.

V. EXPERIMENTS AND ANALYSIS OF ACCURACY

As in [1], all PIMs, including the new reconfigurable models were implemented using the MPI library of C in Linux, and for the sake of comparison, experiments were executed on a computational platform using two Xeon E5-2620 2.4 GHz six core processors with hyper-threading.

The basis for comparing the performance of PIMs, are sequential versions of GA, GAD, DE and PSO with populations of size \( 24n \log n \) and breeding cycles fixed as \( n \). Also, we select eight 12-island asynchronous HoPIMs, designed in [12], running exclusively one of the BAs: GA, GAD, DE or PSO. Furthermore, we select two asynchronous HePIMs designed in [1]. The homogeneous models are \( P^{Tr12A}_{Tr12A} \), \( P^{Tr12A}_{Tr12A} \), \( P^{Tr12A}_{Tr12A} \), \( P^{Pso12A}_{Pso12A} \), \( P^{Pso12A}_{Pso12A} \), \( P^{Gad12A}_{Gad12A} \) and \( P^{Gad12A}_{Gad12A} \). The superscripts denote the BA used by the homogeneous model. The subscript prefixes denote whether the model uses the static tree (\( Tr \)) or the dynamic complete graph topology (\( gbmm \), and the subscript suffix 12A indicates the number of islands and that the model is asynchronous. From [1], we select the heterogeneous PIMs \( P^{GA}_{Tr12A} \) and \( P^{GA}_{gbmm12A} \), the latter provided the best quality results.

A. Parameter Setup

We use the parameters for BAs, HoPIMs and HePIMs obtained in [1]. The parameter tuning adopted the “taxonomy T1” in [31]. Tab.I presents the parameter ranges. For percentages, the tested values range between 2% and 100%. For probabilities, the values range from 0 to 1, and for the mutation parameter from 0.01 to 0.02. For DE, the \( F_M \) parameter ranges from 1% to 2% since values above 2% degrade the quality of solutions. For PSO, the parameters to guide the particles in the search space are self-adjusting.

The setup tested all models over packages of twenty \( n \)-gene permutations, \( n \in \{50, 60, \ldots, 140, 150\} \). All parameters were evaluated, and those values that provided the best solutions were selected (see Tabs. II and III). For HePIMs, only the migration parameters were calibrated using the same evolutionary parameters as HoPIMs. Reconfigurable frequency is added to reconfigurable HePIMs (see Tab.IV).
In contrast, PSO experiments, shows that the better the accuracy. The radar chart in Fig. 4, from previous since URD is a minimization problem, the smallest the result, computed number of reversals for each unsigned permutation. each permutation is taken as the result. The average gives the permutations of size \( n \) for the dynamic complete graph topology, the best solutions were computed by \( P_{\text{recHet}}^{\text{gbmm12A}} \) and the worst by \( P_{\text{Gbmm12A}}^{\text{gbmm12A}} \). In contrast to the fact that GAD provided better accuracy than GA, the homogeneous models \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) respectively outperformed \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) and \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \).

Table VI exemplify the final island configuration. We ran \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) and \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) over one hundred entries of size 100 and computed the average of the final distribution of the four BAs over the islands. Surprisingly the proportion of islands running GA is dominant for \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) that can be explained since the final average results of sets of islands with the same BA are very similar for this model (see right chart on Fig. 5). On the other side, the distribution BAs over islands for \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) is better balanced (cf. right chart on Fig. 6).

B. Analysis of Accuracy

HePIMs use parameters taken from Tabs. II, III and IV according to the parameter setting obtained in [1]. Reconfiguration frequency (RF) percentages of 14% and 24% delivered better results in the new reconfigurable HePIMs \( P_{\text{recHet}}^{\text{Gbmm12A}} \) and \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \), respectively; this gives respectively seven and four reconfiguration cycles.

For each permutation size, \( n \in \{100, 110, \ldots, 150\} \), one package of one hundred unsigned permutations with \( n \) genes was randomly generated. All PIMs were executed ten times (a total of one thousand executions) using each one of the \( n \) permutations of size \( n \), and the average of these executions for each permutation is taken as the result. The average gives the computed number of reversals for each unsigned permutation.

Non- and reconfigurable HePIMs accuracies are compared. Since URD is a minimization problem, the smallest the result, the better the accuracy. The radar chart in Fig. 4, from previous experiments, shows that DE is the best adapted BA for the URD problem [1]. In contrast, PSO provided the worst results, while GA and GAD, in this order, gave competitive results. The six radii of the chart represent the accuracy for inputs of sizes 100, 110 to 150. The ranges and scales in each radius of the radar chart are adjusted to simplify the presentation.

PIMs with the tree and the complete graph topologies outperformed, as expected, their sequential versions [1]. The radar charts to the left in Figs. 5 and 6 show that the HoPIMs maintained the order of competitiveness of their best and worst adapted BAs: the best quality solutions were obtained by \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) and the worst by \( P_{\text{Gbmm12A}}^{\text{Gbmm12A}} \) for the static tree model, while for the dynamic complete graph topology, the best solutions
each set of three islands in the HePIMs ($P^{\text{het}}_{\text{B12A}}$ and $P^{\text{recHet}}_{\text{B12A}}$, respectively) that execute the same BA. These charts make it evident that the migration policy and application of diverse BAs in the islands of the heterogeneous architectures successfully propagates the results obtained in all islands, but are not enough to outperform the quality of the homogeneous model running the best adapted BA, $P^{\text{Het}}_{\text{B12A}}$ and $P^{\text{recHet}}_{\text{B12A}}$, respectively.

The performance of reconfigurable PIMs is included in Figs. 5 and 6. The model $P^{\text{recHet}}_{\text{B12A}}$ computed better quality results than the pure heterogeneous model $P^{\text{Het}}_{\text{B12A}}$, and closed the gap between this and the best-adapted homogeneous architecture $P^{\text{recHet}}_{\text{B12A}}$, running DE (see radar chart on the left in Fig. 5). It makes evident that adding the versatility of reconfiguration to HePIMs improves their performance. On the other side, the reconfigurable HePIM $P^{\text{recHet}}_{\text{B12A}}$ computed quality results that are indistinguishable from the competitive ones of $P^{\text{recHet}}_{\text{B12A}}$.

Fig. 7 compares the accuracy of the non- and reconfigurable HePIMs: $P^{\text{recHet}}_{\text{B12A}}$, $P^{\text{recHet}}_{\text{B12A}}$, and $P^{\text{recHet}}_{\text{B12A}}$. From the experiments, it is clear that the new reconfigurable HePIMs add to the versatility of heterogeneous PIMs the flexibility of dynamically updating the BA executed in each island, promoting in this manner not only data diversity but also algorithmic dynamism. Reconfigurable HePIMs open up a promising new exploration space where unlike von Newman’s style of algorithmic exploration focused on efficient information data management, algorithmic data dynamism enters as a crucial player in the game (e.g., [32], [33]).

Although this work aims to improve the accuracy of solutions, it is important to stress that the new reconfigurable HePIMs maintain speedups competitive regarding the HePIMs, as shown in Table V. HoPIMs’ speedups are computed regarding their sequential versions, and non- and reconfigurable HePIMs’ speedups are regarding the running time average of the sequential algorithms. The reconfiguration overhead can be measured as the speedup ratio of reconfigurable to non-reconfigurable HePIMs: 0.87 for $P^{\text{recHet}}_{\text{B12A}}$, and 0.88 for $P^{\text{recHet}}_{\text{B12A}}$.

C. Statistical Analysis

Statistical tests validated experiments using 95% of significance level represented in the tests as $\alpha = 0.05$. The samples are the sets of one hundred outputs obtained in Section V-B. Initially, the Friedman test was applied to define the control algorithm. Then, Holm’s test was applied to check the null hypothesis that the performance of the control algorithm is the same concerning the remaining algorithms, according to García and Herrera approach [34] (see also [35], and [36]).

Table VII presents the statistical results for the PIMs with binary tree topology: $P^{\text{WT}}_{\text{B12A}}$, $P^{\text{recHet}}_{\text{B12A}}$, and $P^{\text{recHet}}_{\text{B12A}}$. The Friedman test selects $P^{\text{recHet}}_{\text{B12A}}$ as the control algorithm, and the null hypothesis is discarded for all samples. The model $P^{\text{recHet}}_{\text{B12A}}$ is the second-best confirming the discussion in Section V-B.

Table VIII shows the statistical results for the models with complete graph topology, $P^{\text{recHet}}_{\text{B12A}}$ and the best HoPIM, $P^{\text{Het}}_{\text{B12A}}$, selected as the control algorithm. Finally, Table IX gives the statistical tests for the four reconfigurable and non-reconfigurable HePIMs. The selected control algorithm was $P^{\text{Het}}_{\text{B12A}}$. Holm’s procedure rejects the null hypotheses (for $p$-value $\leq 0.05$), thus, $P^{\text{recHet}}_{\text{B12A}}$ has statistical significance only for $P^{\text{recHet}}_{\text{B12A}}$, and $P^{\text{recHet}}_{\text{B12A}}$, and compared to $P^{\text{recHet}}_{\text{B12A}}$, there is no statistical significance confirming the discussion in Section V-B.

VI. Conclusions and Future Work

Reconfigurable heterogeneous PIMs that dynamically update different bio-inspired algorithms on their islands were introduced. Two reconfigurable HePIMs with different archipelago-topologies were designed: a static binary tree topology, $P^{\text{recHet}}_{\text{B12A}}$, and a dynamic complete-graph topology, $P^{\text{recHet}}_{\text{B12A}}$. The asynchronous models ran four BAs in their islands: GA, GAD, PSO, and DE. The new reconfigurable HePIMs were tested over the unsigned reversal distance problem and with a small overhead outperformed the result quality of associated non-reconfigurable HePIMs, closing the gap with the homogeneous best-adapted model that uses DE.
Experiments evaluated statistically confirmed the benefits of reconfigurable HePIMs. Such reconfigurable models preserve the power of HePIMs to navigate efficiently in the space of feasible solutions through a healthy balance between individual and island diversity and migration policy. In addition, the novel reconfiguration feature increases HePIMs’ flexibility by allowing PIMs to evolve dynamically. Thus, improving PIMs’ algorithmic adaptability to solve the target problem.

Future work will explore reconfigurable HePIMs over different problems and with a greater variety of BAs. In particular, we believe that the dynamic algorithmic heterogeneity promoted by the new model will help deal with multiobjective optimization problems. Indeed, the reconfiguration would encourage the application of the best-adapted BA to each target problem over the islands.

REFERENCES

[1] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón, “Heterogeneous Parallel Island Models,” in IEEE Symposium Series on Computational Intelligence (SSCI), 2021.
[2] J. H. Holland, “Genetic Algorithms and the Optimal Allocation of Trials,” SIAM Journal on Computing, vol. 2, no. 2, pp. 88–105, 1973.
[3] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” J. of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997.
[4] R. Eberhart and J. Kennedy, “Particle swarm optimization,” in IEEE Int. Conf. on neural networks, vol. 4, pp. 1942–1948, 1995.
[5] K. A. D. Jong and W. M. Spears, “A formal analysis of the role of multi-point crossover in genetic algorithms,” Annals of Mathematics and Artificial Intelligence, vol. 5, p. 1–26, 1992.

Fig. 5. (a) $P_{\text{recHet}}^{\text{Tr12A}}$, $P_{\text{Het}}^{\text{Tr12A}}$ and related HoPIMs accuracy; (b) $P_{\text{recHet}}^{\text{Tr12A}}$, $P_{\text{Het}}^{\text{Tr12A}}$ accuracy and average of each set of islands in $P_{\text{Het}}^{\text{Tr12A}}$ running the same BA.

Fig. 6. (a) $P_{\text{recHet}}^{\text{gbmm12A}}$, $P_{\text{Het}}^{\text{gbmm12A}}$ and related HoPIMs accuracy; (b) $P_{\text{recHet}}^{\text{gbmm12A}}$, $P_{\text{Het}}^{\text{gbmm12A}}$ accuracy and average of islands in $P_{\text{Het}}^{\text{gbmm12A}}$ running the same BA.
Fig. 7. Accuracy of the non reconfigurable and reconfigurable HePIMs

| L. | Control | t | Algorithm | \( p \)-value | \( \text{es} / \%) |
|---|---|---|---|---|---|
| 100 | gbmm12A | 3 | \( T^0_{T12A} \) | 0.629498386201959E-7 | 0.016 |
| 100 | gbmm12A | 3 | \( P^0_{T12A} \) | 3.484903855850034E-7 | 0.025 |
| 110 | gbmm12A | 1 | \( P^0_{T12A} \) | 0.395980957118269 | 0.05 |
| 110 | gbmm12A | 3 | \( P^0_{T12A} \) | 2.144216625367106E-13 | 0.016 |
| 110 | gbmm12A | 2 | \( T^0_{T12A} \) | 7.21791651482415E-18 | 0.025 |
| 120 | gbmm12A | 1 | \( P^0_{T12A} \) | 0.170903522037079 | 0.05 |
| 130 | gbmm12A | 2 | \( P^0_{T12A} \) | 3.9085576772719E-7 | 0.025 |
| 130 | gbmm12A | 1 | \( P^0_{T12A} \) | 0.721258402177081 | 0.05 |
| 140 | gbmm12A | 2 | \( P^0_{T12A} \) | 3.41551806462443E-13 | 0.025 |
| 150 | gbmm12A | 1 | \( P^0_{T12A} \) | 0.411317917762579 | 0.05 |
| 150 | gbmm12A | 3 | \( P^0_{T12A} \) | 2.144216625367106E-13 | 0.016 |
| 150 | gbmm12A | 2 | \( P^0_{T12A} \) | 4.3425992230487E-19 | 0.016 |
| 150 | gbmm12A | 1 | \( P^0_{T12A} \) | 2.432421662536713E-16 | 0.025 |
| 150 | gbmm12A | 3 | \( P^0_{T12A} \) | 2.89924985256243E-19 | 0.016 |
| 150 | gbmm12A | 2 | \( P^0_{T12A} \) | 4.949796820984E-15 | 0.025 |

[6] A. J. Umbarkar and D. P. Sheth, “Crossover operators in genetic algorithms: a review,” ICTACT J. on Soft Comp., vol. 6, no. 1, 2015.
[7] A. Caprara, “Sorting by reversals is difficult,” in Proc. of the first ACM annual Int. Conf. on Comp. molecular Biology, pp. 75–83, 1997.
[8] 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.
[9] J. Kecskeméti and D. Sankoff, “Exact and approximation algorithms for the inversion distance between two chromosomes,” in Proc. Combinatorial Pattern Matching (CPM), pp. 87–105, Springer, 1993.
[10] T. G. Crainic and M. Toulouse, “Parallel Genetic Algorithms for Mixed Integer Problems,” in Multiple-Objective Optimization: Algorithms and Applications, vol. 475–513, Boston, MA: Springer US, 2003.
[11] G. R. d. C. L. Duarte, A. C. da Fonseca, L. G. de Lima, and B. S. L. Pires, “An Island Model based on Stigmergy to solve optimization problems,” in Proc. of the first ACM annual Int. Conf. on Evol. Comp. (CEC), 2018.
[12] L. A. da Silveira, J. L. Soncco-Alvarez, T. A. de Lima, and M. Ayala-Rincón, “Parallel Island Model Genetic Algorithms Applied in NP-Hard problems,” in Proc. of the first ACM annual Int. Conf. on Evol. Comp. (CEC), 2019.
[13] Y. Zhang, D. Gong, and Z. Ding, “Handling multi-objective optimization problems with a multi-swarm cooperative particle swarm optimizer,” Expert Systems with Applications, vol. 38, pp. 13933–13941, 2011.
[14] B. Xu, Y. Zhang, D. Gong, Y. Guo, and M. Rong, “Environmental Sensitivity-Based Cooperative Co-Evolutionary Algorithms for Dynamic Multi-Objective Optimization,” IEEE/ACM Trans. on Comp. Biology and Bioinformatics, vol. 15, no. 6, pp. 1877–1900, 2018.
[15] D. Gong, B. Xu, Y. Zhang, Y. Guo, and S. Yang, “A Similarity-Based Cooperative Co-Evolutionary Algorithm for Dynamic Interval Multiobjective Optimization Problems,” IEEE Trans. on Evol. Comp., vol. 24, no. 1, pp. 142–156, 2020.
[16] R. Hashimoto, H. Ishibuchi, N. Masuyama, and Y. Nojima, “Analysis of Evolutionary Multi-Tasking as an Island Model,” in Proc. of the Genetic and Evol. Comp. Conf. (GECCO), pp. 1894–1897, ACM, 2018.
[17] A. Eiben and S. Smit, “Parameter tuning for configuring and analyzing evolutionary algorithms,” Swarm and Evol. Comp., vol. 1, no. 1, pp. 19–31, 2011.
[18] R. W. Hartenstein, “Reconfigurable Computing: boosting software education for the multicore era: Why we need to reinvent computing,” in 11th Int. Conf. on Intell. Systems Design and Applications, pp. 265–270, IEEE, 2011.
[19] K. Yasuda and K. Yazawa, “Parameter self-adjusting strategy for particle swarm optimization,” in Proc. of the first ACM annual Int. Conf. on Evol. and Swarm Intelligence Algorithms, pp. 483–491, 2001.
[20] L. A. da Silveira, J. L. Soncco-Alvarez, and M. Ayala-Rincón, “Parallel multi-island genetic algorithm for sorting unsigned genomes by reversals,” in Proc. of the first ACM annual Int. Conf. on Evol. Comp. (CEC), 2018.
[21] P. V. S. Z. Capriles, L. G. Fonseca, H. J. C. Barbosa, and A. C. C. Lemos, “Rank-based ant colony algorithms for truss weight minimization with discrete variables,” Communications in Numerical Methods in Engineering, vol. 23, no. 6, pp. 553–575, 2007.
[22] T. G. Crainic and M. Toulouse, “Parallel Genetic Algorithms on Distributed-Memory Architectures,” in NATUG-6: Proc. of the Sixth Conf. of the North American Transputer Users Group on Transputer Research and Applications, pp. 39–47, 1991.
[23] S. C. Lin, W. F. Punch, and E. D. Goodman, “Coarse-grain parallel genetic algorithms: categorization and new approach,” in Proc. of the sixth IEEE Symp. on Parallel and Distributed Processing, pp. 28–37, 1994.
[24] D. Izzo, M. Rucinski, and C. Ampatzis, “Parallel global optimisation meta-heuristics using an asynchronous island-model,” in IEEE Cong. on Evol. Comp. (CEC), pp. 2301–2308, 2009.
[25] Y. Gong and A. Fukunaga, “Distributed island-model genetic algorithms using heterogeneous parameter settings,” in IEEE Cong. on Evol. Comp. (CEC), pp. 820–827, 2011.
[26] G. Diarte, A. Lemonge, and L. Goliatt, “A New Strategy to Evaluate the Attractiveness in a Dynamic Island Model,” in IEEE Cong. on Evol. Comp. (CEC), 2018.